Grafica al calcolatore

-

Computer Graphics

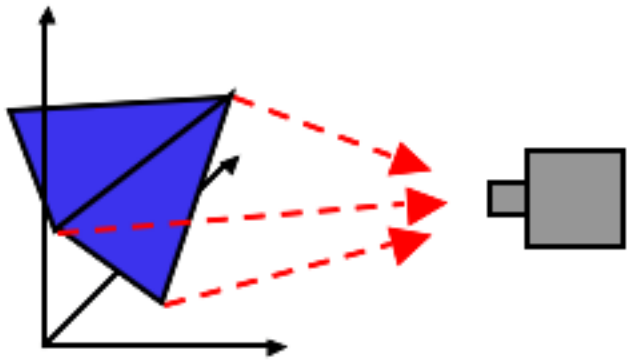
7 – Pipeline di rasterizzazione

Rasterization pipeline

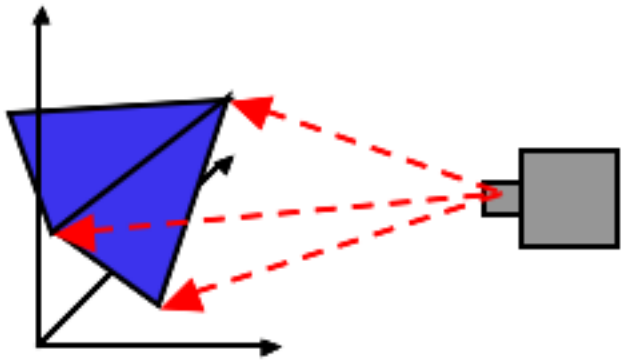
- Sappiamo implementare ray casting (o ray tracing).
- Abbiamo tuttavia già visto che le applicazioni grafiche usano un metodo diverso. Perché?
  - Anche il solo ray casting con un modello di illuminazione locale `e troppo oneroso in termini computazionali, a causa dei calcoli delle intersezioni raggi-oggetti (per non parlare del ray tracing)
- Si usa un diverso paradigma: la rasterizzazione.
- Non più modelli generici, ma maglie poligonali, le altre rappresentazioni si possono ricondurre a questa.
  - ray casting era trasversale rispetto alla modellazione della scena

Rasterization pipeline

- Invece che gettare (to cast) raggi sulla scena, proietto la scena (composta di poligoni) sul piano immagine.
- La proiezione di un poligono è ancora un poligono, che ha per vertici le proiezioni dei vertici
- La rasterizzazione è un esempio di rendering in object-order, mentre ray-casting è un esempio di rendering in image-order..



Rasterization:
Project geometry forward



Ray casting:
Project image samples backwards

Problemi

- Eliminato il calcolo delle intersezioni raggio-oggetto. Ma ho molti altri problemi:
  - Superfici nascoste: nel ray casting il problema di stabilire quali parti della scena sono visibili e quali nascoste viene implicitamente risolto: è la prima intersezione lungo il raggio che conta. Nella rasterizzazione, quando proietto i poligoni, devo stabilire quali sono visibili (e dunque da disegnare) e quali no.
  - Clipping: nel ray casting gli oggetti al di fuori del volume di vista vengono implicitamente scartati. Nella rasterizzazione devo stabilire esplicitamente quali poligoni entrano nel volume di vista e quali no. Quelli a cavallo vanno tagliati.
  - Scan-conversion: la proiezione dei poligoni sul piano immagine non tiene conto della natura discreta dell'immagine stessa. Alla fine devo disegnare un poligono su una matrice di pixel, decidendo dunque quali pixel vi appartengono e quali no.

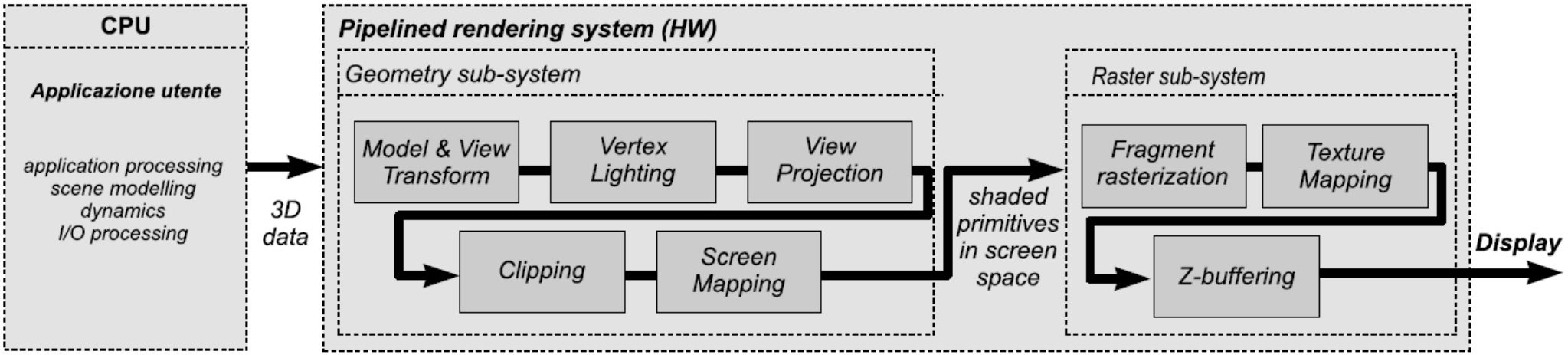
Problemi

- Shading interpolativo: Nel ray casting il colore di ogni pixel deriva dall'applicazione del modello di illuminazione al corrispondente punto della scena. Nel nuovo paradigma gli unici punti collegati esplicitamente a punti della scena sono i vertici dei poligoni. Per questi ultimi si può assegnare un colore, e per quelli interni bisogna interpolare.
- Effetti globali di illuminazione: Nel ray casting era facile ottenere le ombre tracciando gli shadow rays. Più in generale, con il ray-tracing si possono ottenere effetti di illuminazione globale che invece nella rasterizzazione richiedono trucchi più o meno elaborati (oppure se ne fa a meno).

- La soluzione di questi problemi è implementata nella pipeline in modo efficiente e la rasterizzazione risulta più veloce del ray casting.

Pipeline di rasterizzazione

- Avevamo già visto questo schema
  - Si parte da una lista di primitive (triangoli)
  - Si operano una serie di processi a catena (pipeline), parte in CPU, parte in GPU
  - Pipeline divisa in due parti: la prima geometrica, la seconda "raster" che lavora sui pixel dell'immagine (la vera rasterizzazione)

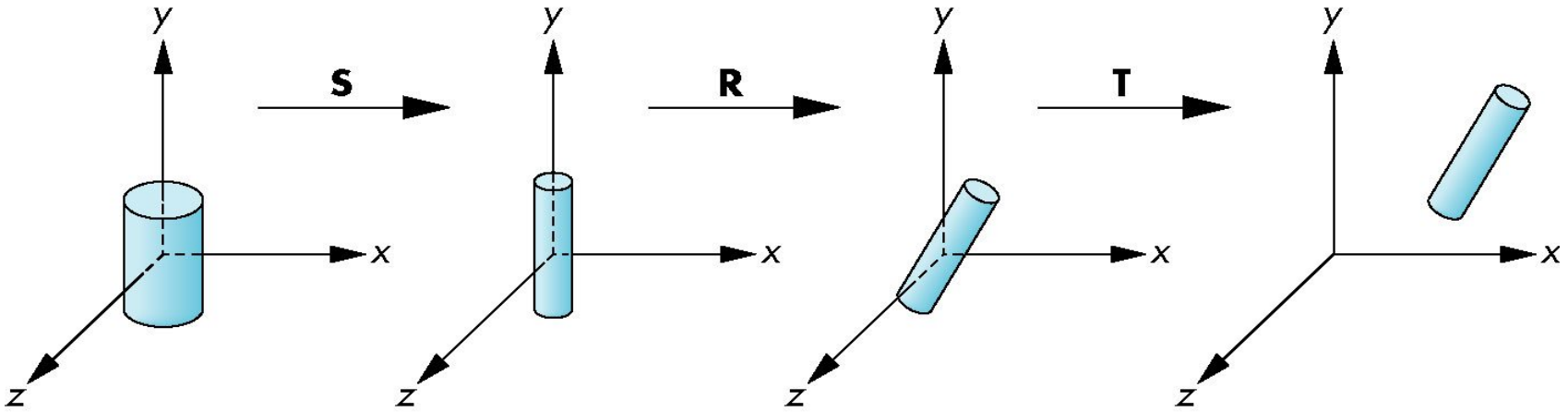


Implementazione ed evoluzione

- In OpenGL possiamo considerare la pipeline grafica come una macchina a stati ove si definiscono delle primitive e delle operazioni che le modificano
- Il programma definisce queste e le passa alla pipeline che dà in ouput le immagini
- Primitive di due tipi: geometriche e raster
- Vedrete in pratica le specifiche della pipeline di openGL
  - Vediamo ora quali sono le operazioni base per realizzare la visualizzazione di una scena triangolata

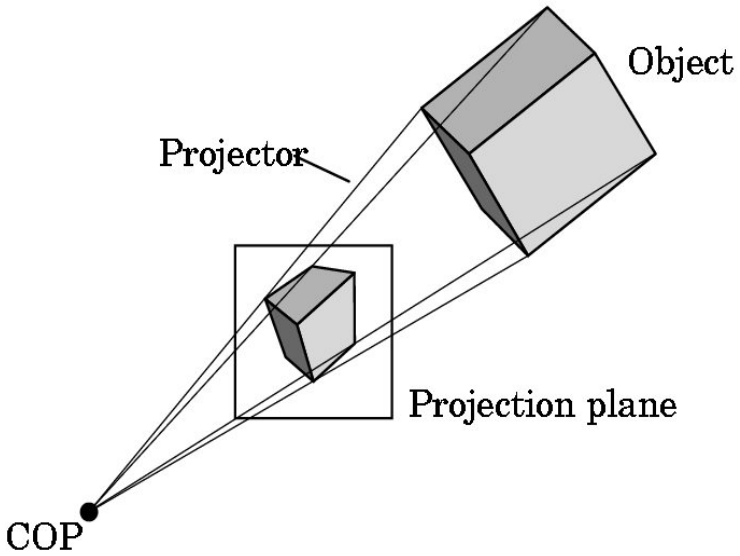
Pipeline geometrica

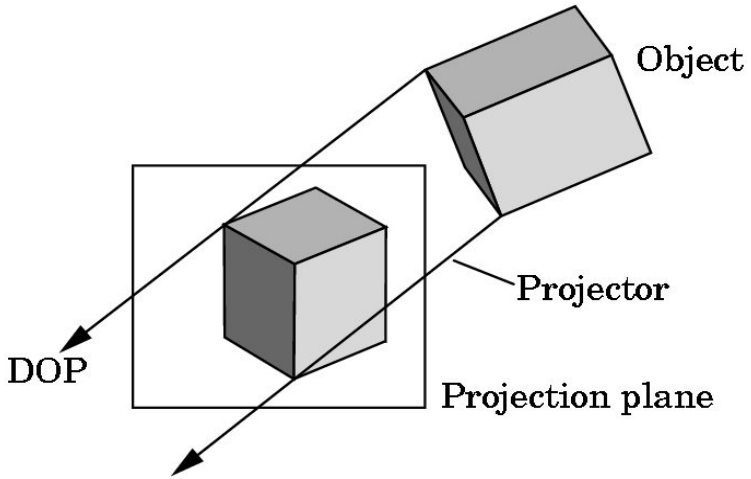
- Devo
  - mettere assieme gli oggetti della scena
  - rappresentare gli oggetti nel sistema di riferimento della telecamera virtuale
  - Proiettarli
  - Posso applicare le varie trasformazioni geometriche



Proiezione

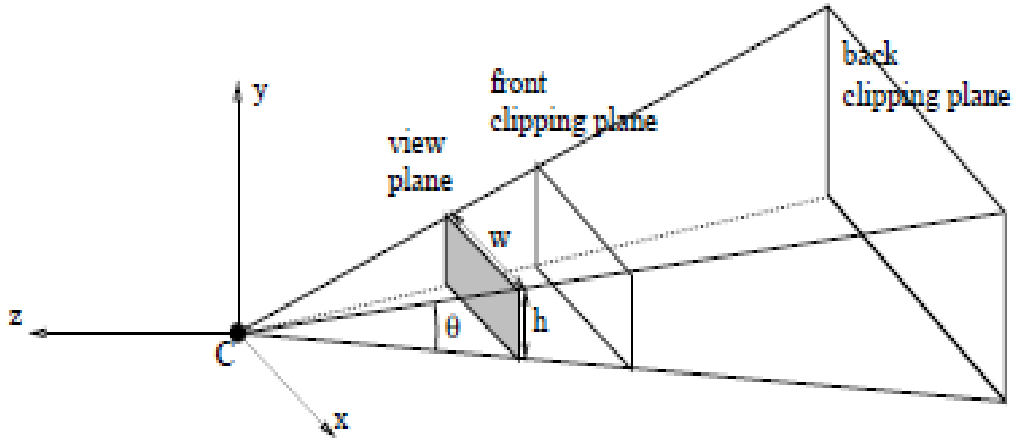
- Quindi metto tutti gli oggetti con trasformazioni "di modellazione" (traslazioni, rotazioni, scalature) in un riferimento "coordinate mondo"
- Poi devo proiettare sul piano immagine
  - Proiezione prospettica o affine





Trasformazione di vista

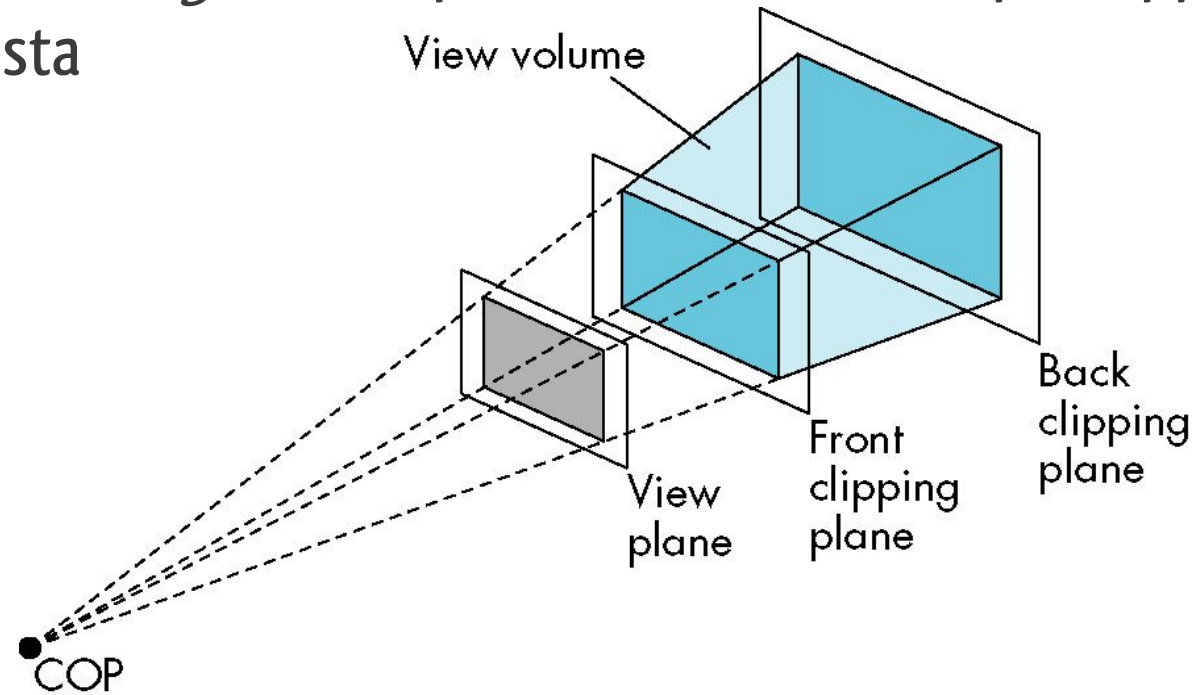
- Supponiamo di usare la prospettica.
- Devo passare in coordinate solidali con la camera (view space, come in figura) e definire cosa "vedo" dato che lavorando in object order devo buttare via il resto (clipping)



- Convenzioni
  - Il piano immagine è davanti alla telecamera a distanza 1 (d era solo un fattore di scala globale).
  - L'angolo di vista (solido) si assegna mediante l'angolo di vista (verticale) θ ed il fattore di aspetto a = w/h della finestra di vista.

Volume di vista

- La piramide di vista, in principio semi-infinita, viene limitata da due piani paralleli al piano di vista: il piano di taglio anteriore (front clippig plane) e quello posteriore (back clipping plane).
  - Il solido risultante è un frustum, e prende il nome specifico di frustum di vista (view frustum), o volume di vista (view volume).
  - Il frustum di vista è la regione di spazio nel mondo che può apparire sulla finestra di vista



Volume di vista

- Gli oggetti che cadono dentro il volume di vista vengono disegnati.
- Per proiettarli con M o $(\mathrm{M}_{\perp})$ le coordinate devono essere trasformate nello spazio vista, con una trasformazione rigida, detta trasformazione di vista.
- Si noti che abbiamo messo il piano vista davanti al centro di proiezione, ma l'asse Z punta "indietro", quindi la matrice di proiezione rimane la stessa vista prima (con d = 1):

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Trasformazione prospettica

- La proiezione prospettica M potrebbe essere applicata ai punti P (vertici dei triangoli) espressi nello spazio vista ottenendo le coordinate sul piano immagine
- Invece si fa una cosa più contorta: si applica la "trasformazione prospettica", che mappa il frustum in un parallelepipedo retto (volume di vista canonico), poi mappato sul piano immmagine con una proiezione ortografica.
- Motivo: semplificare le operazioni di clipping (buttar via ciò che sta fuori)

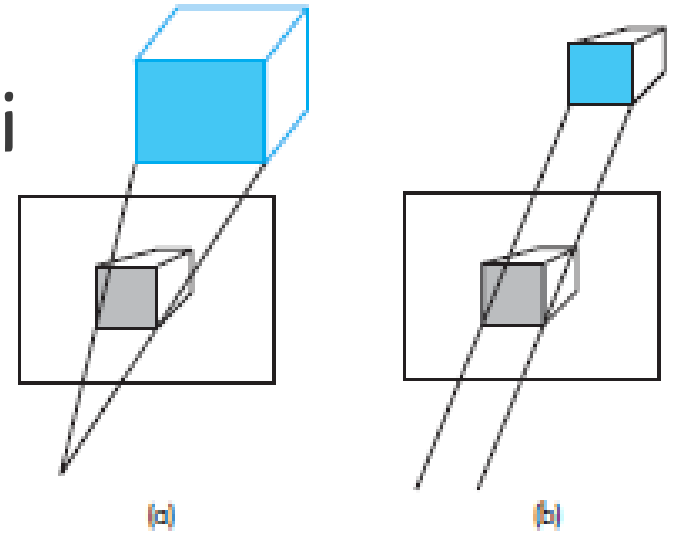


FIGURE 4.23 Predistortion of objects. (a) Perspective view. (b) Orthographic projection of distorted object.

Trasformazione prospettica

- Consideriamo la matrice 4 × 4 N, non singolare (simile alla M) (che prende il nome di matrice di matrice di trasformazione o di normalizzazione prospettica).

$$N = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \alpha & \beta \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

- Applicando N a P si ottiene la 4-pla $(x, y, \alpha z+\beta, -z)$ che, dopo la divisione prospettica diventa $(-x/z, -y/z, -\alpha-\beta/z)$
- Le prime due componenti sono identiche alla proiezione standard, ma la terza componente (pseudo-profondità) è diversa: $z_s = \alpha z+\beta, -z$.
- per valori di α,β fissati è una funzione monotona di z. La relazione tra z e z_s è non lineare, ma l'ordinamento sulla profondità è conservato.

Volume di vista canonico

La trasformazione (normalizzazione) prospettica N mappa punti 3D in punti 3D, e scegliendo opportunamente α, β possiamo fare in modo che mappi il frustum di vista in un parallelepipedo con coordinate arbitrarie. Gli oggetti vengono distorti di conseguenza.

Proiettando questo parallelepipedo ortogonalmente (eliminando la terza coordinata, cioè z_s) si ottiene la proiezione prospettica desiderata. In OpenGL il volume di vista canonico è un cubo:

$$-1 \le x \le 1 \quad -1 \le y \le 1 \quad -1 \le z \le 1$$

Volume di vista canonico

- Nel volume di vista canonico il back clipping plane ha equazione $z_s = 1$, ed il front clipping plane $z_s = -1$.
- Nel sistema di riferimento camera il front clipping plane di trova a distanza n dall'origine ed il back clipping plane a distanza f
- Vogliamo dunque scegliere α, β in modo che l'intervallo di profondità z $[-n,-f]$ vengamappato in z $[-1, 1]$ (notare l'inversione di z).
- Si ottiene

$$\alpha = \frac{f+n}{n-f} \quad \beta = -\frac{2fn}{n-f}$$

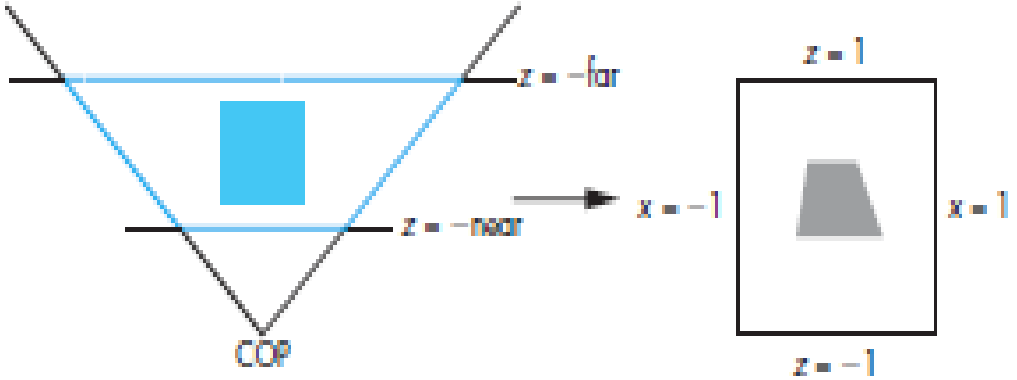


FIGURE 4.39 Perspective normalization of view volume.

Volume canonico

- Se l'angolo di vista $\theta = 90$ ed il fattore d'aspetto a = 1, allora la matrice N con α, β calcolati come sopra trasforma il frustum nel volume di vista canonico
- Nel caso generale, bisogna considerare θ e a, moltiplicando N per la seguente matrice di scalatura, dove $\gamma = 1/\tan(\theta/2)$

che ha l'effetto di trasformare la piramide generica in piramide retta a base quadrata.

$$\begin{pmatrix} \gamma/a & 0 & 0 & 0 \\ 0 & \gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Attenzione: questa matrice agisce anche sulle coordinate x ed y.
- La matrice N viene chiamata anche (in terminologia OpenGL) matrice di proiezione (projection matrix) anche se, a rigore, non effettua una proiezione dello spazio 3D, ma una sua trasformazione.

Caso ortogonale

- Se invece si vuole effettuare una proiezione ortogonale (ortografica), basta sostituire la trasformazione prospettica con una trasformazione (affine) che mappa il volume di vista (un parallelepipedo in questo caso) nel volume di vista canonico.
- Nel caso della proiezione ortografica, il volume di vista è già un parallepipedo, e basta trasformarlo in quello canonico con una opportuna matrice di scalatura.

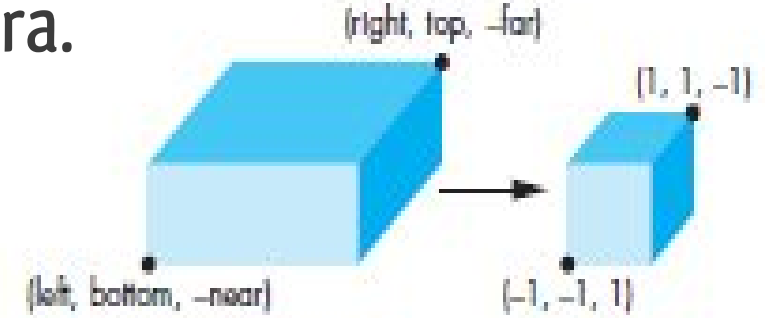
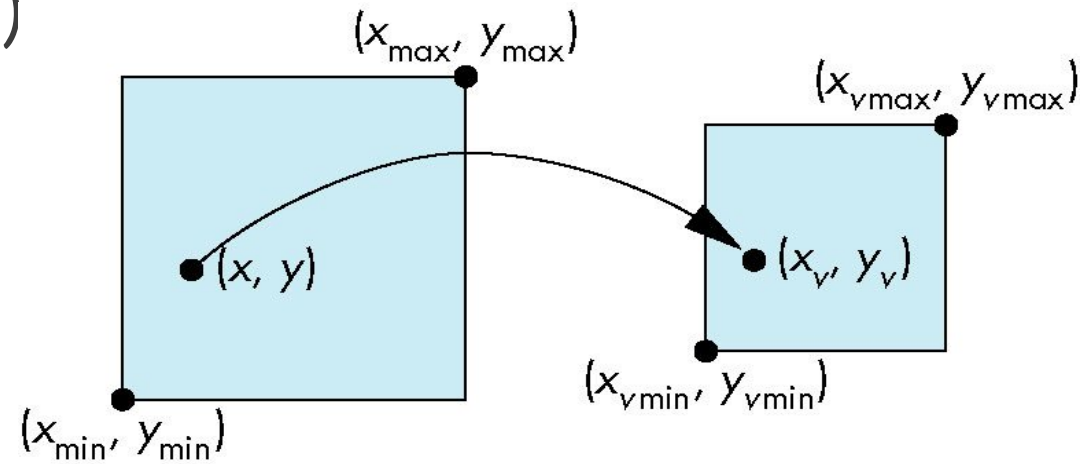


FIGURE 4.25 Mapping a view volume to the canonical view volume.

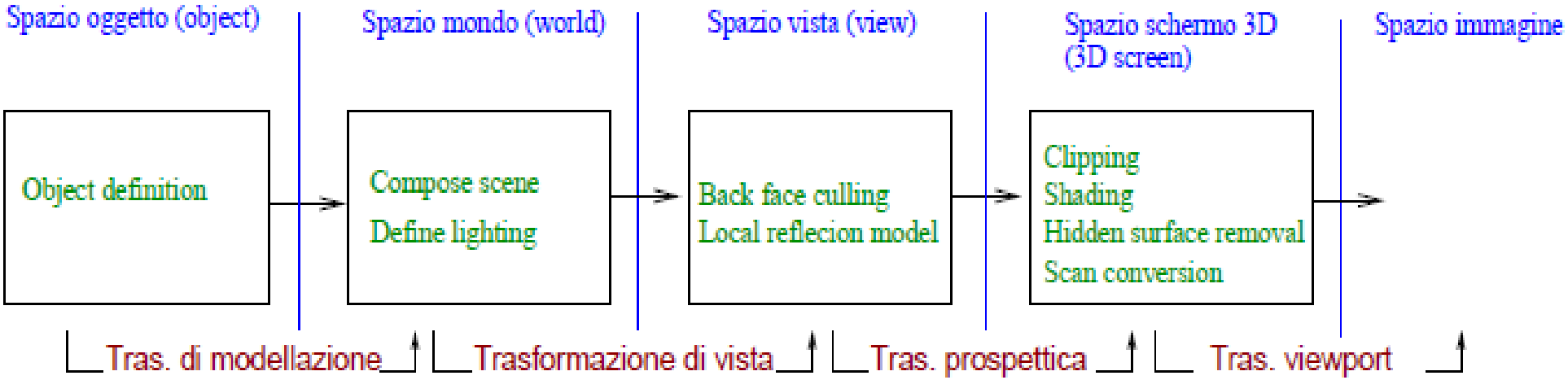
Trasformazione viewport

- Viewport: porzione del display all'interno della quale viene visualizzato il risultato del rendering
- La trasformazione viewport si applica dopo la proiezione ortografica e dipende dalle caratteristiche fisiche del display
- Ai punti proiettati dal volume di vista canonico viene applicata una matrice di trasformazione affine che:
  - ripristina il fattore di aspetto corretto per l'immagine (distorto dalla trasformazione prospettica)
  - scala e trasla l'immagine



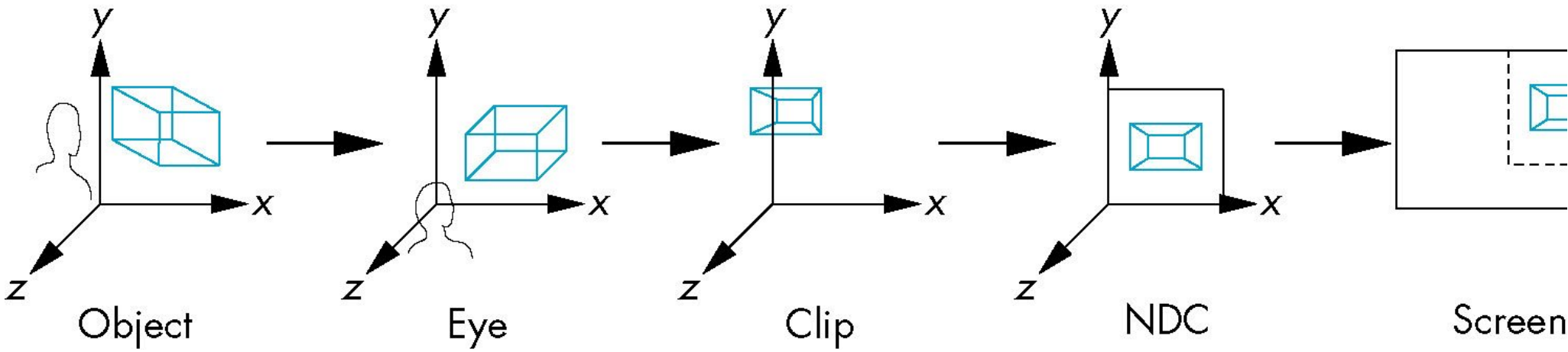
Pipeline geometrica

- Quindi la pipeline geometrica coinvolge diversi sistemi di riferimento e trasformazioni tra di essi.
- In ciascuno di essi avvengono delle operazioni, da parte del codice o della pipeline hardware



Nomi

- I nomi possono variare, ma il senso è quello
- Es da Angel

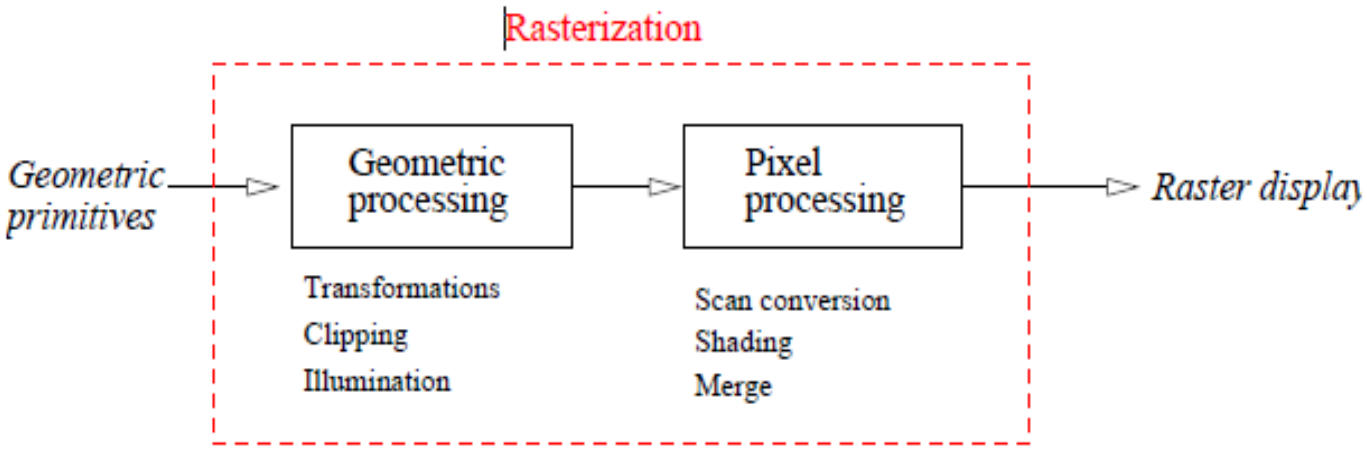


Riepilogo

- Spazio Oggetto (object space): dove ciascun singolo oggetto viene definito. Detto anche local space o modeling space
- Spazio Mondo (world space): dove si costruisce. Si passa dallo spazio oggetto allo spazio mondo mediante la trasformazione di modellazione. Si chiama anche application space.
- Spazio Vista (view space): centrato sulla telecamera virtuale, che definisce, assieme alla finestra di vista ed ai piani di taglio, il frustum di vista. Detto anche camera space o eye space
- Spazio schermo 3-D (3D-screen space): volume di vista canonico, che si ottiene trasformando il frustum di vista in un parallelepipedo. Molte operazioni del processo di rendering avvengono qui. Detto anche 3D normalized device coordinate system (NDC) o normalized projection coordinate system.
- Spazio Immagine (image space): sistema di coordinate nel display fisico (pixel). Si ottiene proiettando ortogonalmente il volume di vista canonico ed applicando la trasformazione di viewport. Si chiama anche (physical) device coordinate system o screen coordinate system.

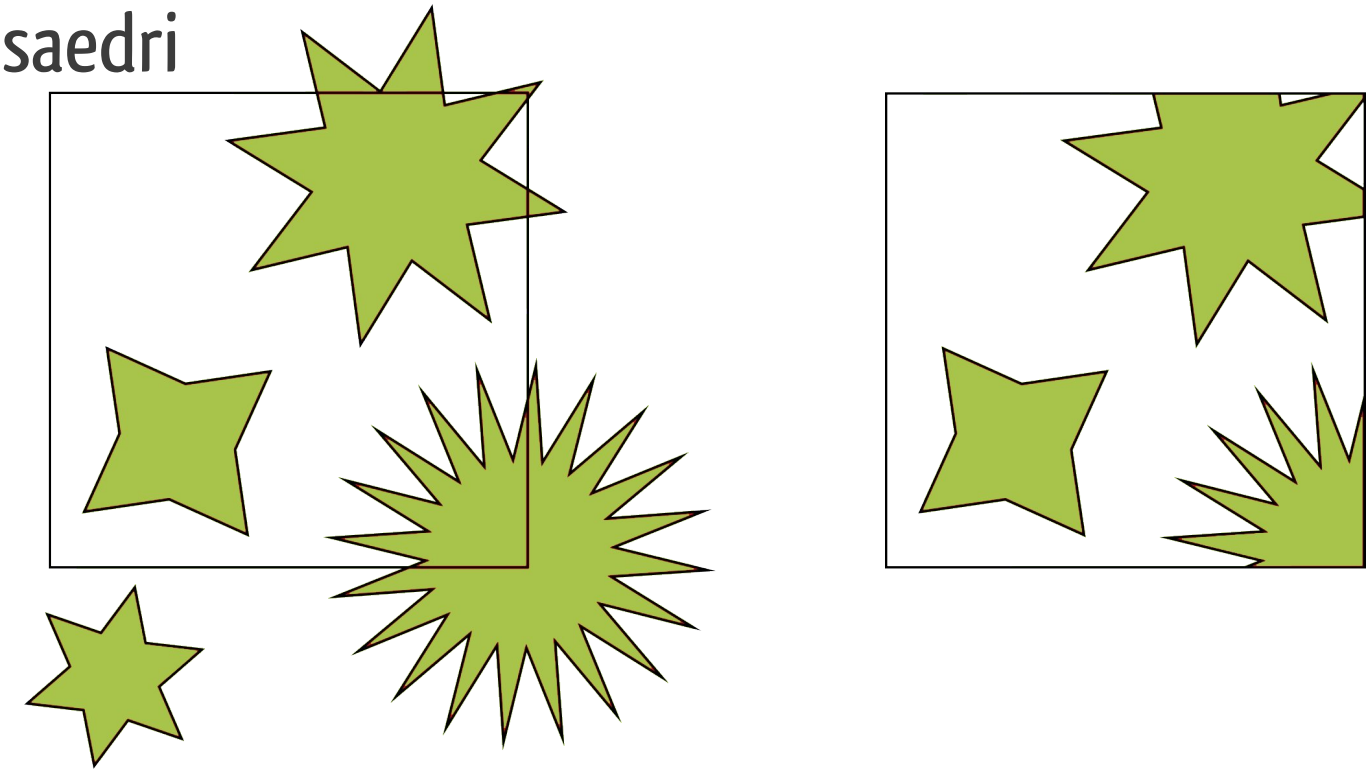
Operazioni

- Clipping: devo eliminare le parti fuori dal frustum e tagliare i poligoni a metà
- Dopo si può procedere a disegnarli sul display (o viewport), generando i frammenti (contributi ai pixel generati dal poligono), assegnando lo shading.
- Utile però semplificare, eliminando superfici nascoste



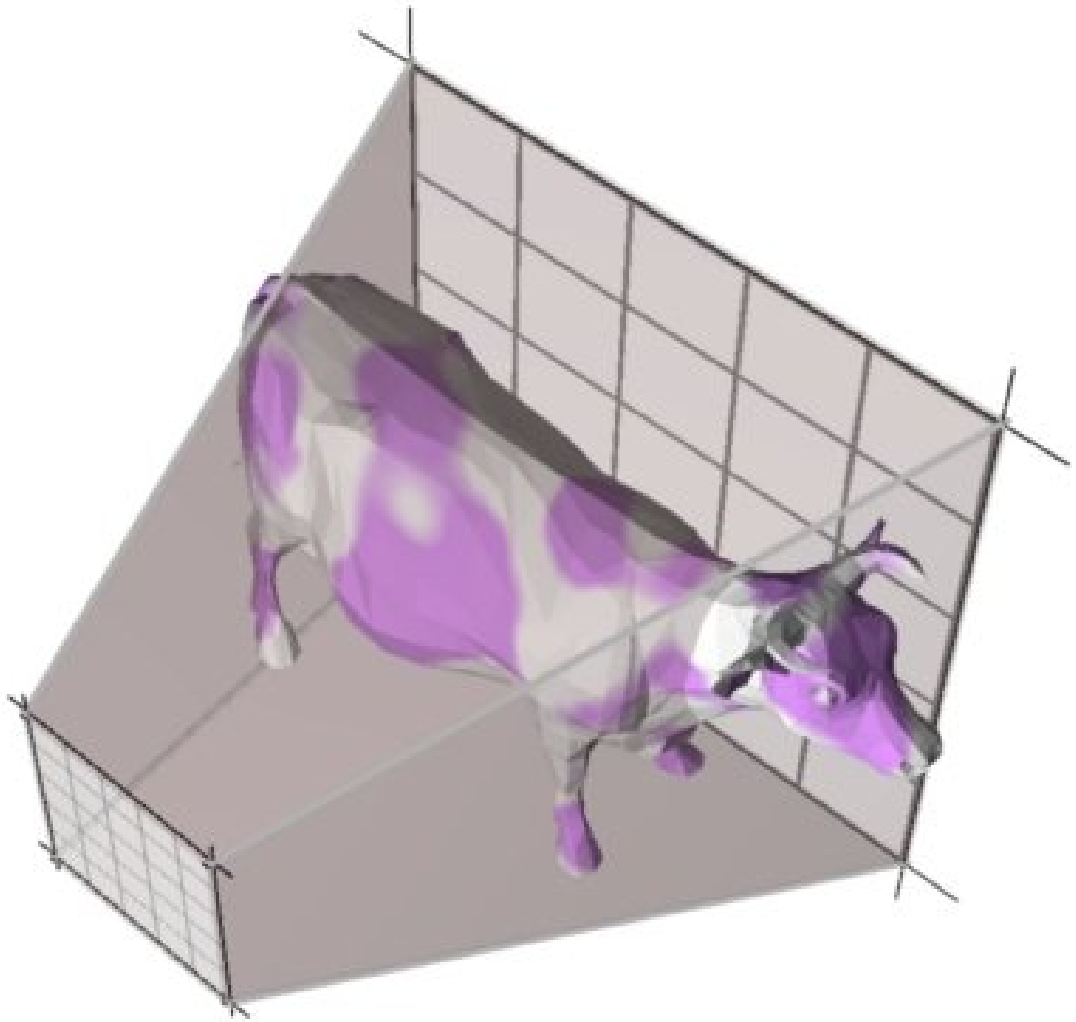
Clipping (generalità)

- L'operazione di clipping consiste nell'individuare (e rimuovere) le primitive grafiche (o parti di esse) esterne ad una finestra rettangolare o esaedrale oppure, più in generale, esterne ad un poligono o poliedro convesso.
- In computer graphics si è interessati al clipping rispetto a rettangoli o esaedri



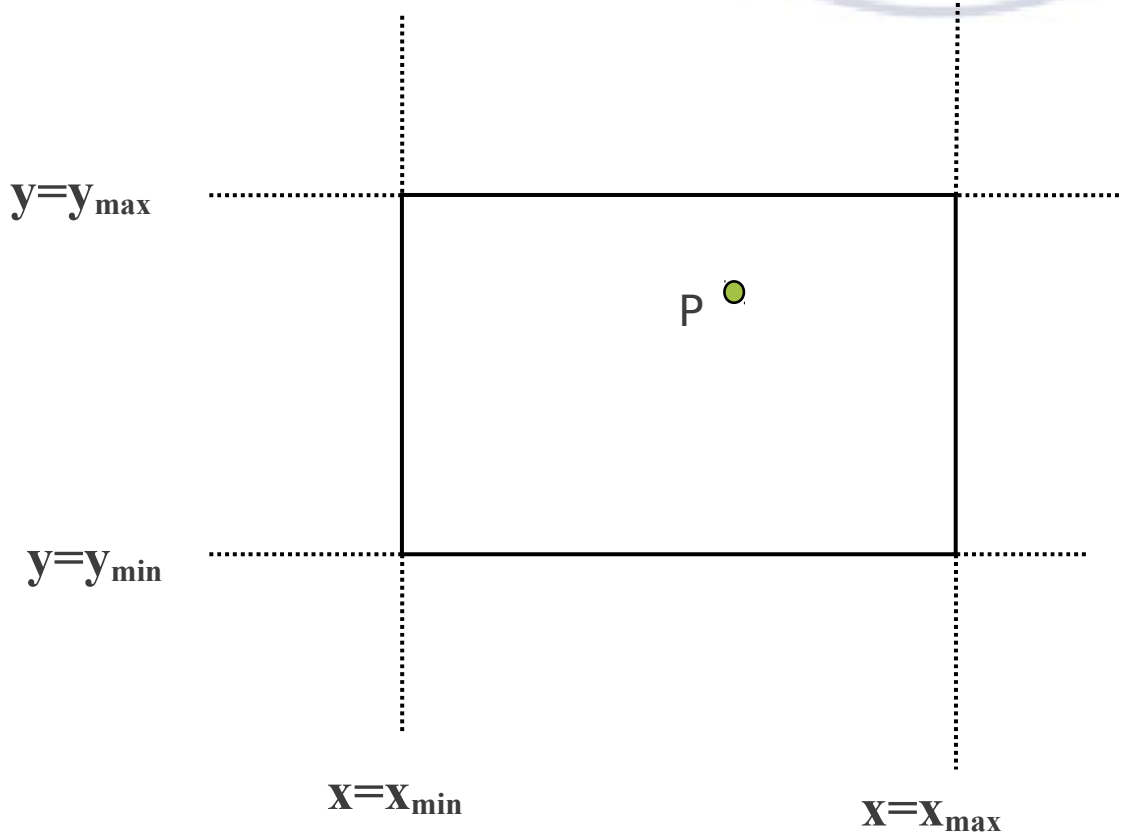
Clipping (generalità)

- Non interessa tutto ciò che non è nel view frustum



Clipping di un punto

- Clipping di un punto: un punto è all'interno del rettangolo di clipping se e solo se sono soddisfatte le 4 disuguaglianze:
  - $x_{\min} \leq x \leq x_{\max}, y_{\min} \leq y \leq y_{\max}$



Clipping di un segmento (basics)

- Clipping di un segmento: necessario analizzare le posizioni dei suoi punti estremi.
  - Se gli estremi sono entrambi interni al rettangolo di clipping, il segmento è interno;
  - Se un estremo è interno e l'altro esterno, allora il segmento interseca il rettangolo di clipping ed è necessario determinare l'intersezione;
  - Se entrambi gli estremi sono esterni al rettangolo, il segmento può intersecare o meno il rettangolo di clipping e si rende necessaria una analisi più accurata per individuare le eventuali parti interne del segmento.

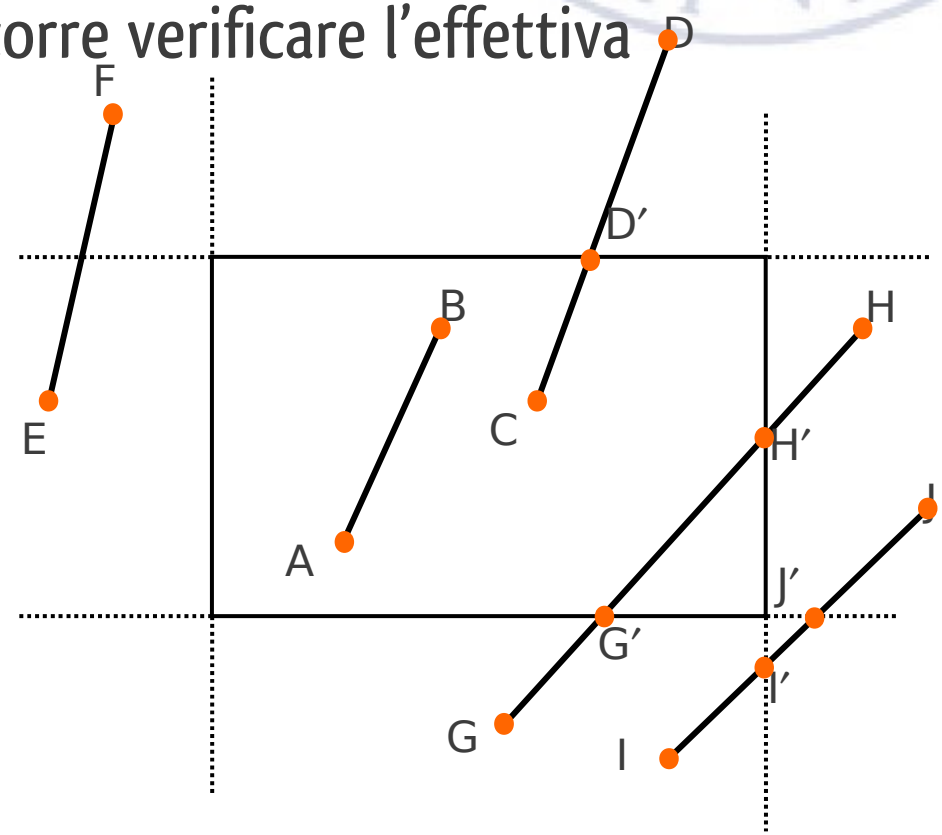
Clipping di un segmento (algoritmo diretto)

L'approccio diretto alla soluzione del problema è quello di determinare le intersezioni tra la retta su cui giace il segmento e le 4 rette su cui giacciono i lati del rettangolo di clipping;

Individuati i punti di intersezione occorre verificare l'effettiva appartenenza al rettangolo di clipping (G′ e H′) o meno (I′ e J′).

Le intersezioni si determinano mediante l'eq. parametrica dei segmenti relativi.

$$x = x_a + (x_b - x_a)t$$
$$y = y_a + (y_b - y_a)t$$
$$t \in [0,1]$$



Clipping di un segmento (algoritmo diretto)

- <u>Approccio diretto</u>
- Per ogni coppia segmento-lato di rettangolo si risolve il sistema di equazioni parametriche che definiscono il segmento in funzione di t_{segm} ed il lato in funzione di t_{lato};
- Se t_{segm} e t_{lato} assumono valori nell'intervallo [0, 1] allora l'intersezione appartiene al segmento ed al rettangolo di clipping;
- E' necessario verificare in anticipo il parallelismo tra le linee prima di determinare l'intersezione;

- Algoritmo costoso e quindi inefficiente.

Clipping di un segmento (Cohen-Sutherland)

Idea di base: le rette che delimitano il rettangolo di clipping suddividono il piano in nove regioni;

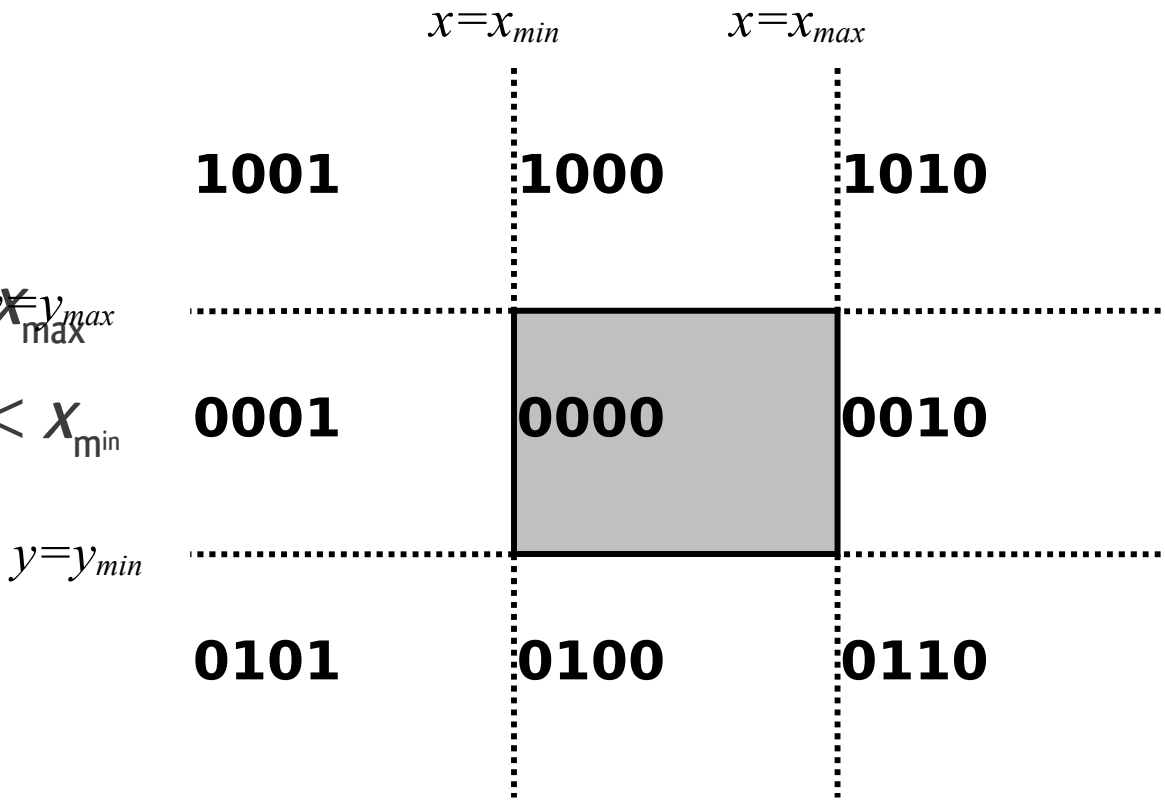
Ad ogni regione viene associato un codice numerico di quattro cifre binarie:

bit 1: sopra edge alto $y > y_{max}$

bit 2: sotto edge basso $y < y_{min}$

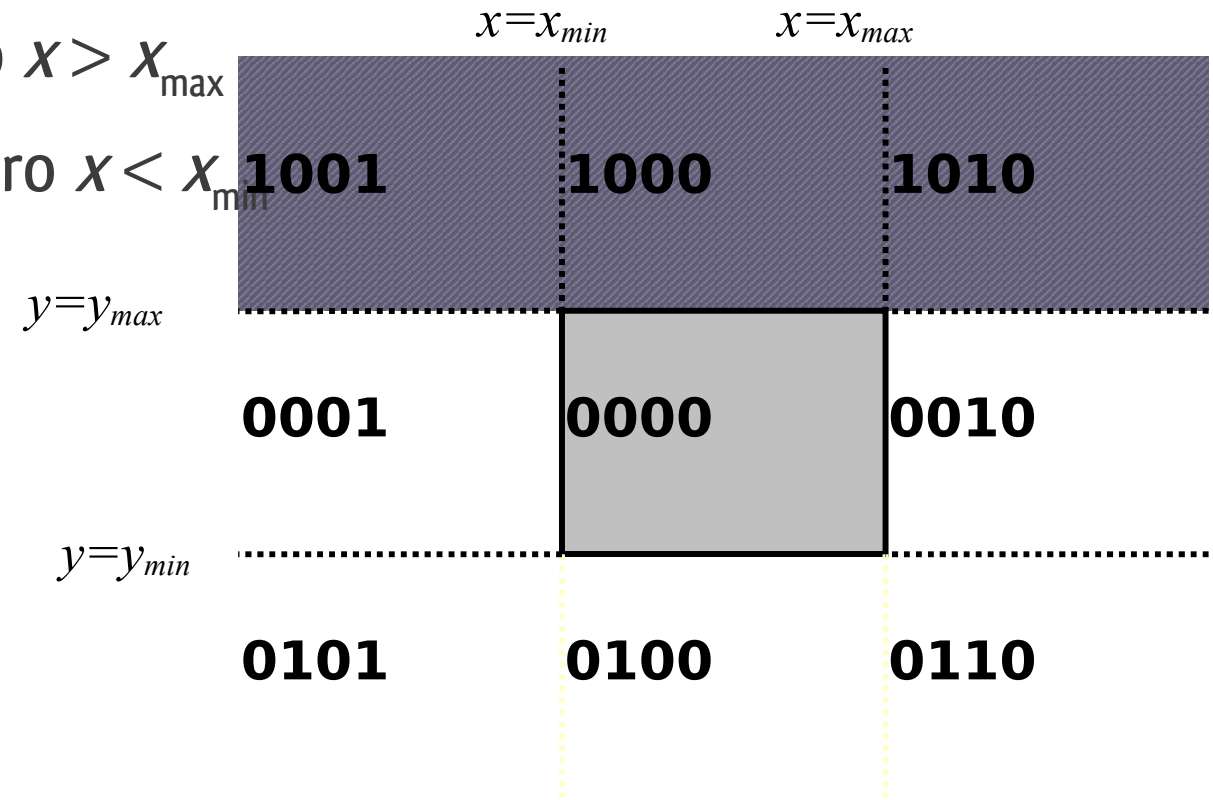
bit 3: a destra edge destro $x > x_{max}$

bit 4: a sinistra edge sinistro $x < x_{min}$



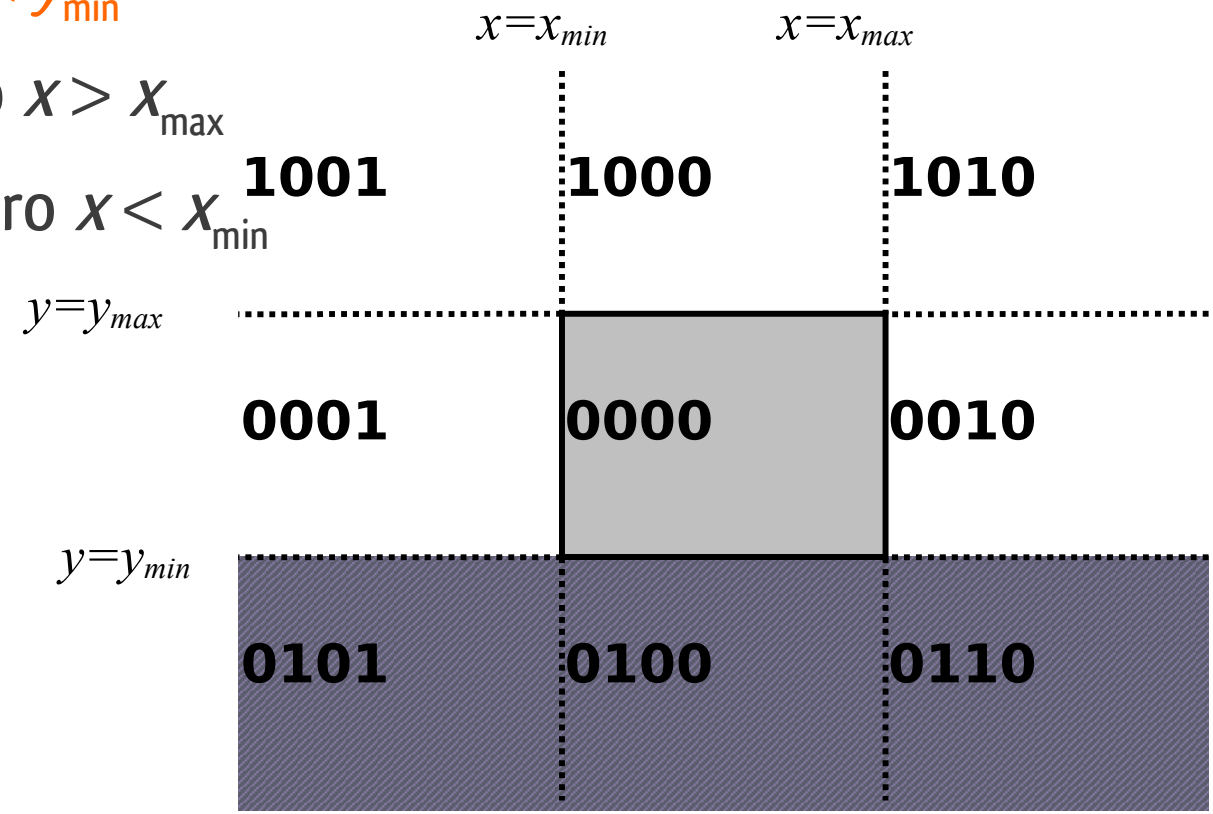
Clipping di un segmento (Cohen-Sutherland)

- Ad ogni regione viene associato un codice numerico di quattro cifre binarie:
- bit 1: sopra edge alto $y > y_{max}$
- bit 2: sotto edge basso $y < y_{min}$
- bit 3: a destra edge destro $x > x_{max}$
- bit 4: a sinistra edge sinistro $x < x_{min}$



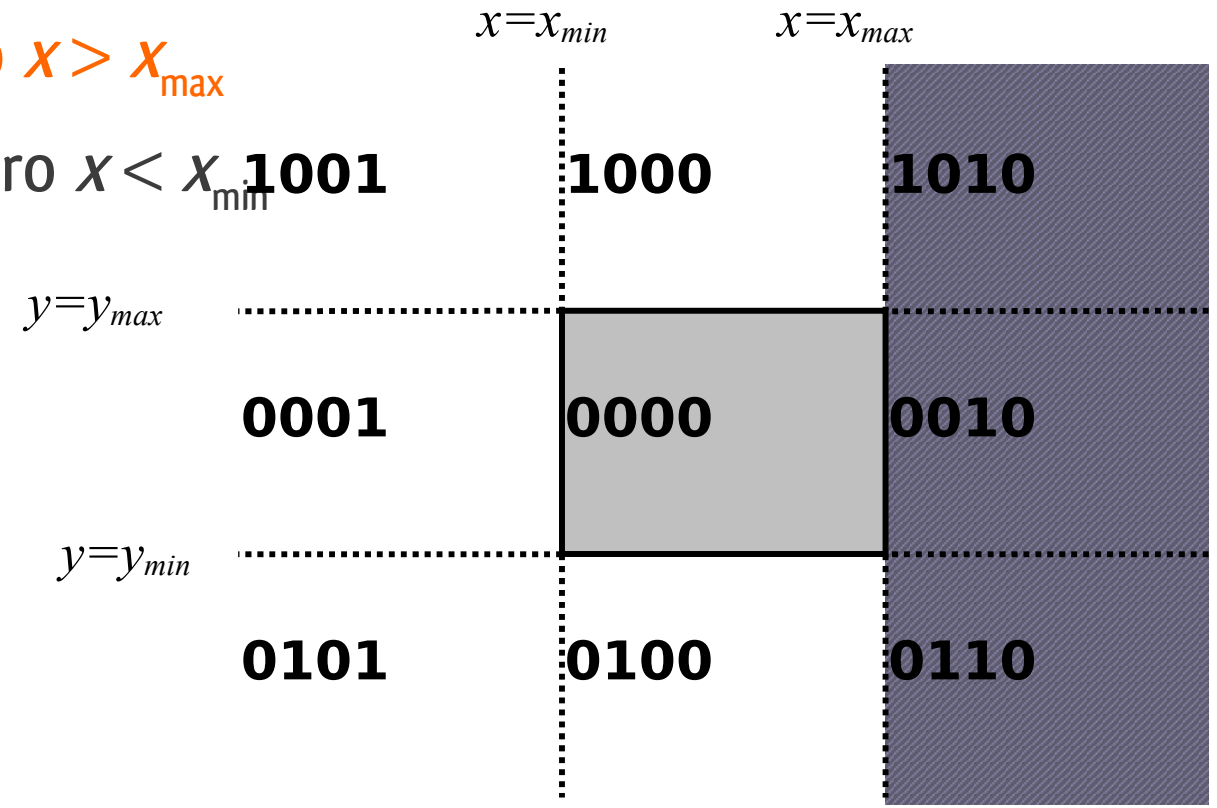
Clipping di un segmento (Cohen-Sutherland)

- Ad ogni regione viene associato un codice numerico di quattro cifre binarie:
- bit 1: sopra edge alto $y > y_{max}$
- bit 2: sotto edge basso $y < y_{min}$
- bit 3: a destra edge destro $x > x_{max}$
- bit 4: a sinistra edge sinistro $x < x_{min}$



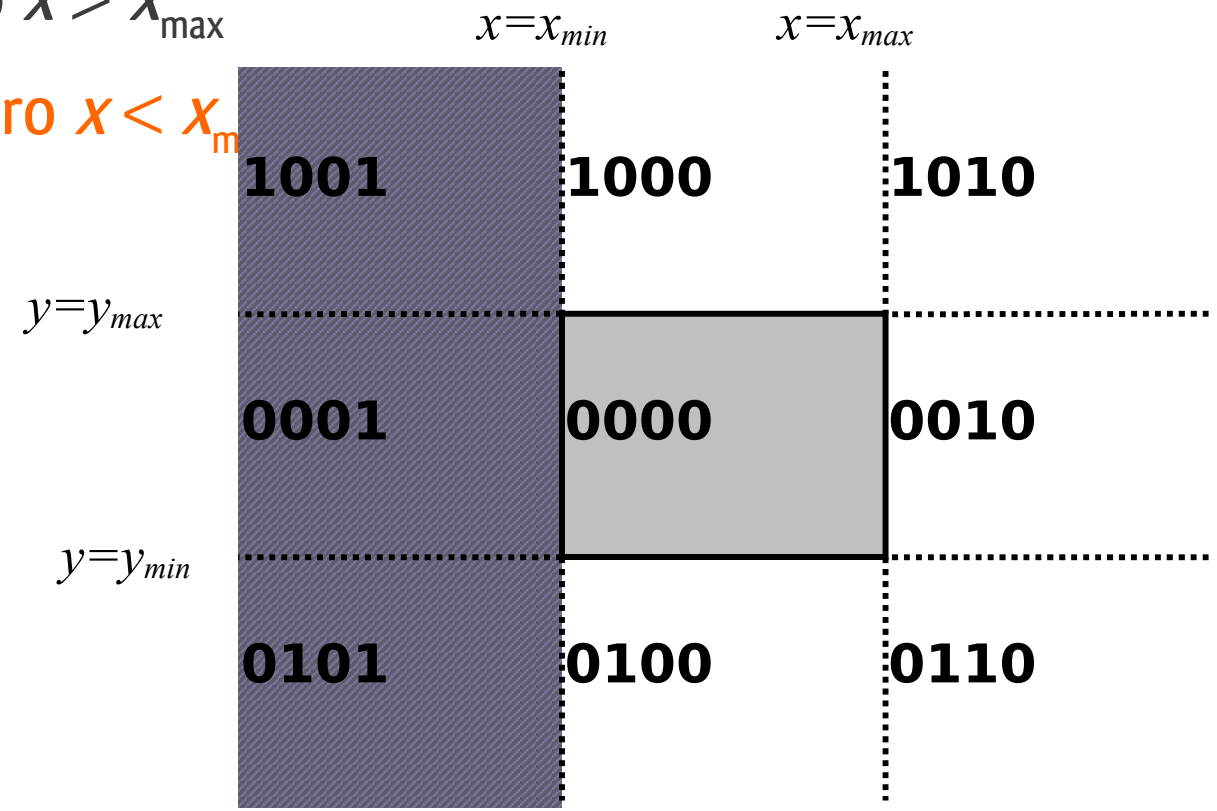
Clipping di un segmento (Cohen-Sutherland)

- Ad ogni regione viene associato un codice numerico di quattro cifre binarie:
- bit 1: sopra edge alto $y > y_{max}$
- bit 2: sotto edge basso $y < y_{min}$
- bit 3: a destra edge destro $x > x_{max}$
- bit 4: a sinistra edge sinistro $x < x_{min}$



Clipping di un segmento (Cohen-Sutherland)

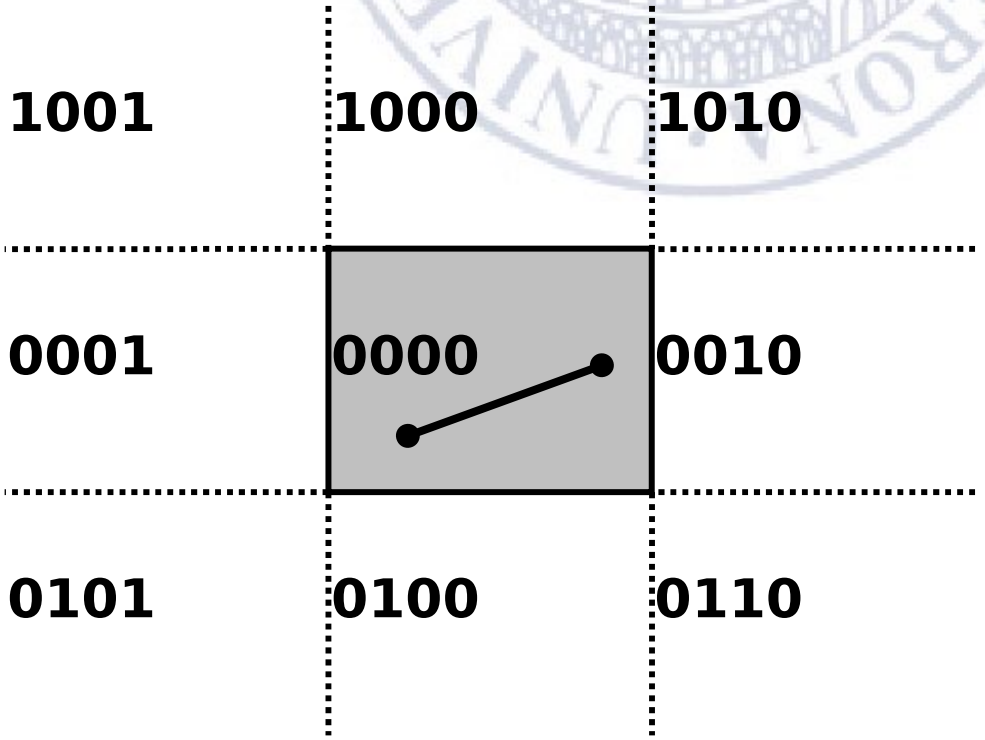
- Ad ogni regione viene associato un codice numerico di quattro cifre binarie:
- bit 1: sopra edge alto $y > y_{max}$
- bit 2: sotto edge basso $y < y_{min}$
- bit 3: a destra edge destro $x > x_{max}$
- bit 4: a sinistra edge sinistro $x < x_{min}$



Clipping di un segmento (Cohen-Sutherland)

- Il clipping di un segmento prevede la codifica (e confronto) dei suoi estremi sulla base delle regioni di appartenenza;

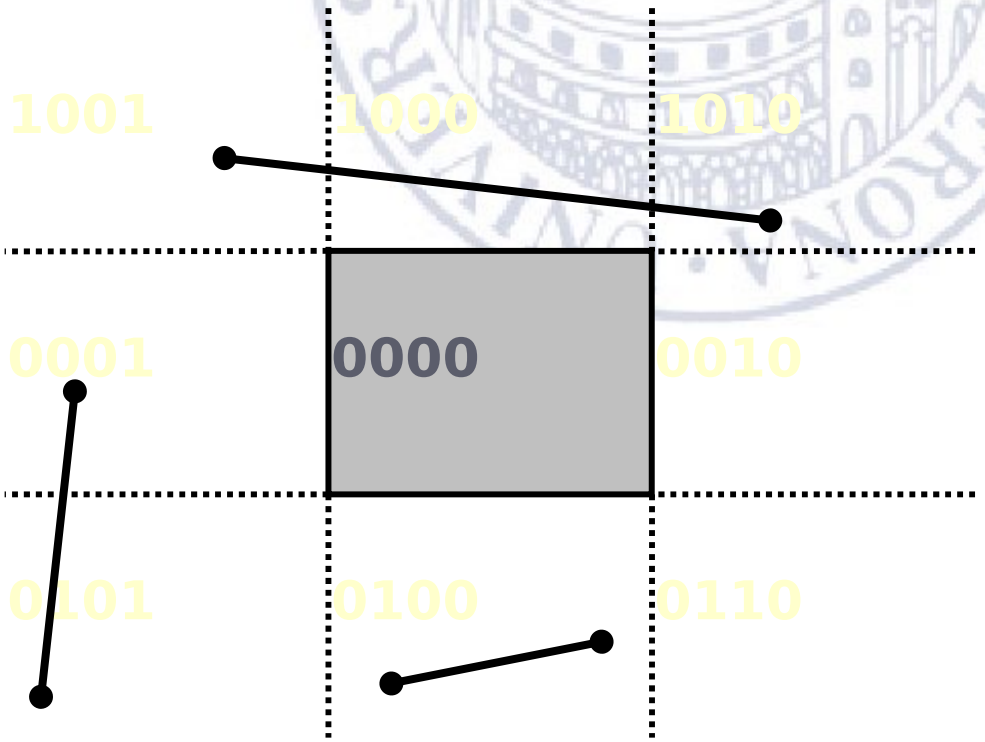
- Se il codice di entrambi gli estremi è 0000 (OR logico tra i codici ritorna un risultato nullo), allora si può banalmente decidere che il segmento è interamente interno al rettangolo di clipping.



$0000 \vee 0000 = 0000$

Clipping di un segmento (Cohen-Sutherland)

- Se l'operazione di AND logico tra i codici degli estremi restituisce un risultato non nullo allora il segmento è esterno al rettangolo di clipping.
- In questo caso, infatti, gli estremi giacciono in uno stesso semipiano (quello identificato dal bit a 1 del risultato) e quindi il segmento non interseca il rettangolo di clipping.



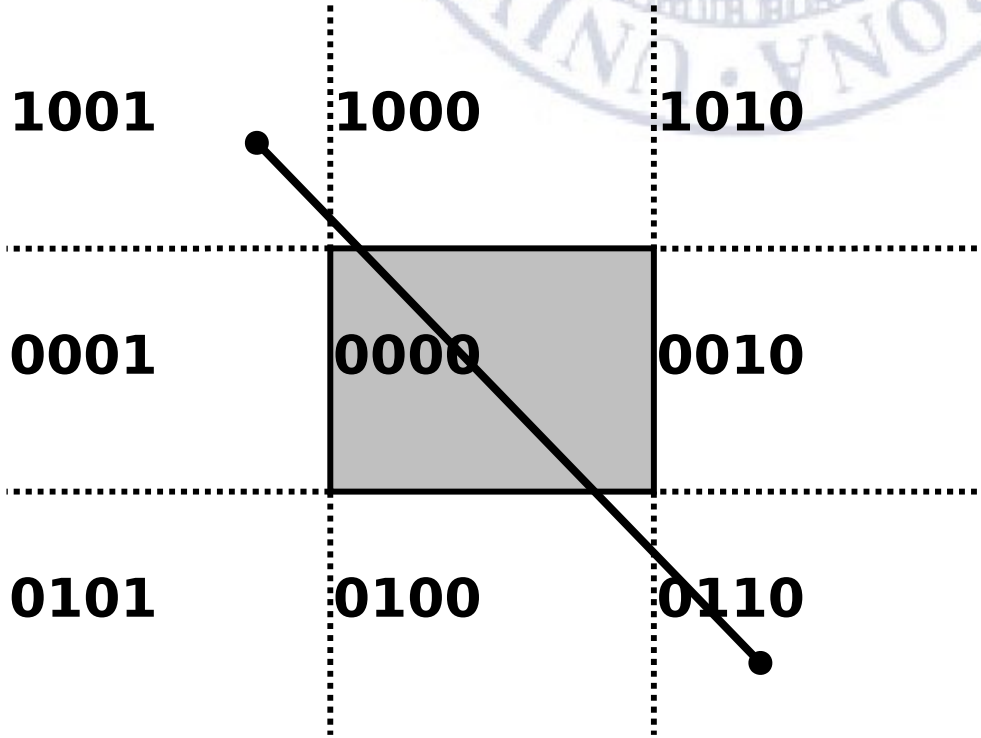
$1001 \wedge 1010 = 1000$

$0100 \wedge 0100 = 0100$

$0001 \wedge 0101 = 0001$

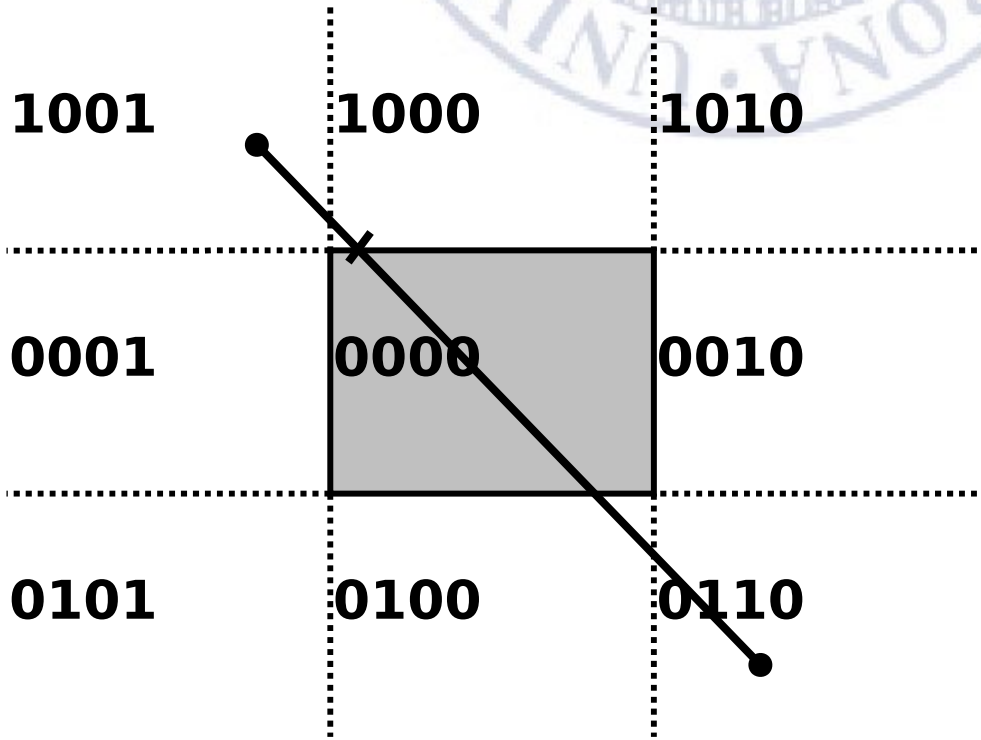
Clipping di un segmento (Cohen-Sutherland)

- Se il risultato dell'AND è nullo:



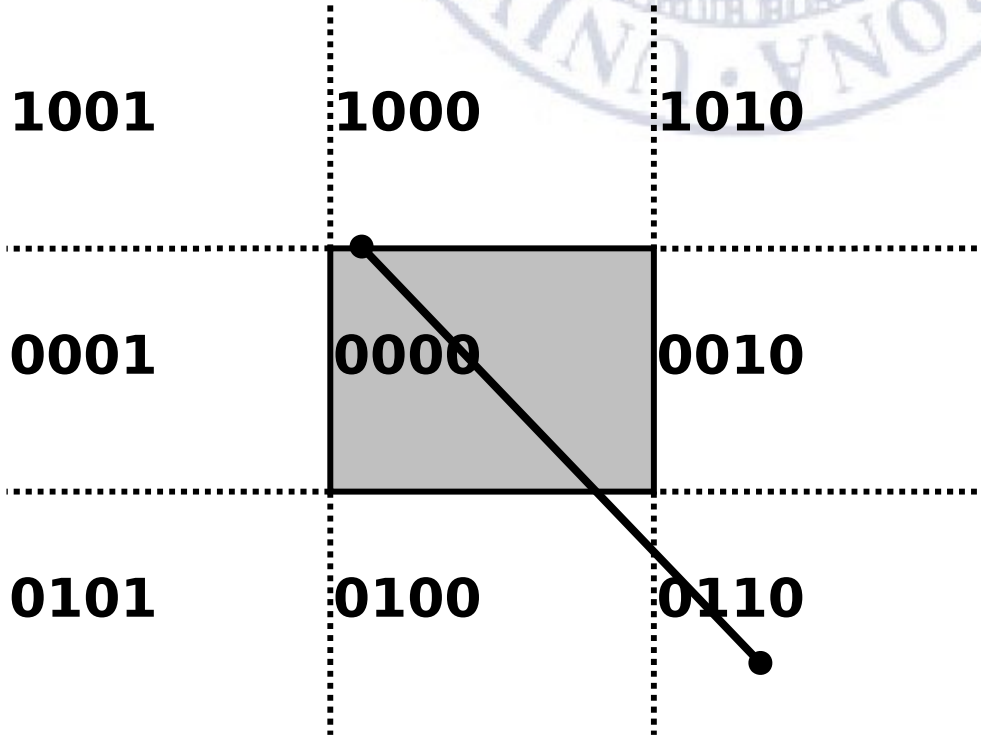
Clipping di un segmento (Cohen-Sutherland)

- Se il risultato dell'AND è nullo:
  - Si individua l'intersezione tra il segmento ed il lato relativo al primo bit discordante tra i codici (bit 1, $y=y_{max}$ in fig.);



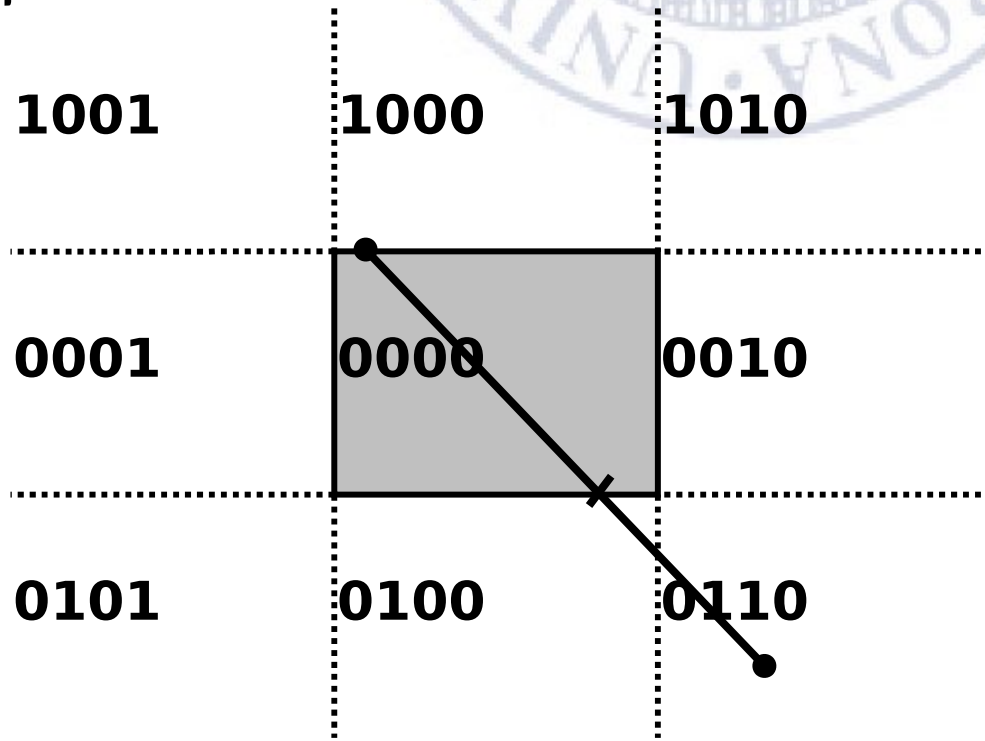
Clipping di un segmento (Cohen-Sutherland)

- Se il risultato dell'AND è nullo:
  - Si individua l'intersezione tra il segmento ed il lato relativo al primo bit discordante tra i codici (bit 1, $y=y_{max}$ in fig.);
  - L'estremo con bit a 1 (in prima posizione nell'esempio) viene sostituito dal nuovo vertice;



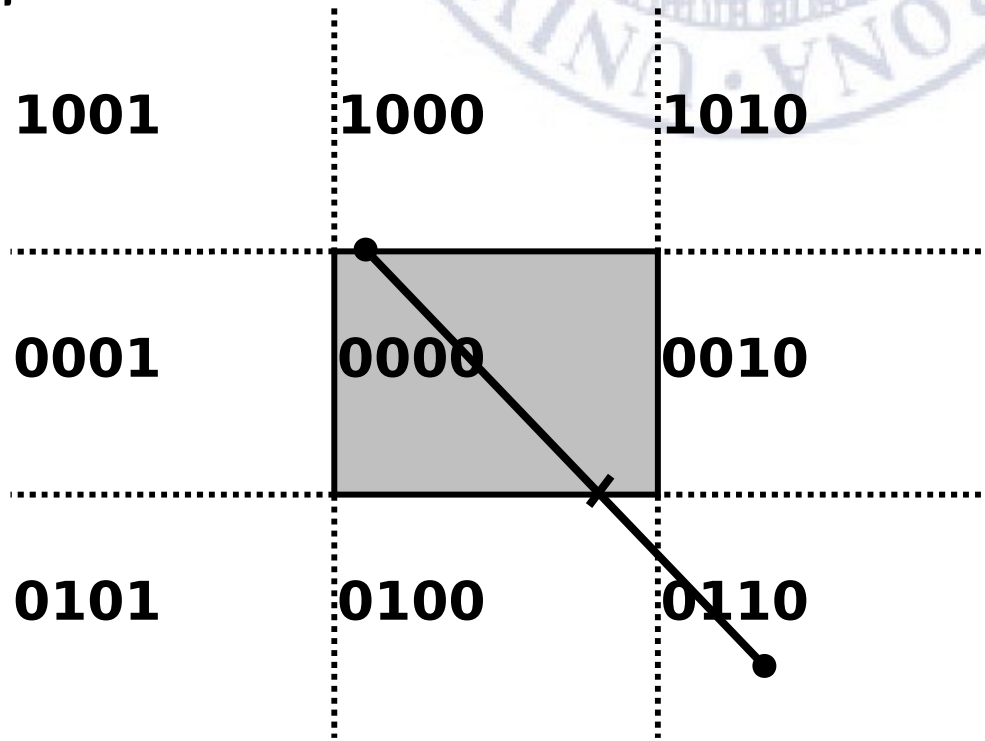
Clipping di un segmento (Cohen-Sutherland)

- Se il risultato dell'AND è nullo:
  - Si individua l'intersezione tra il segmento ed il lato relativo al primo bit discordante tra i codici (bit 1, $y=y_{max}$ in fig.);
  - L'estremo con bit a 1 (in prima posizione nell'esempio) viene sostituito dal nuovo vertice;
  - Si itera il procedimento (in fig., bit 2 discordante, intersezione del segmento con $y=y_{min}$);



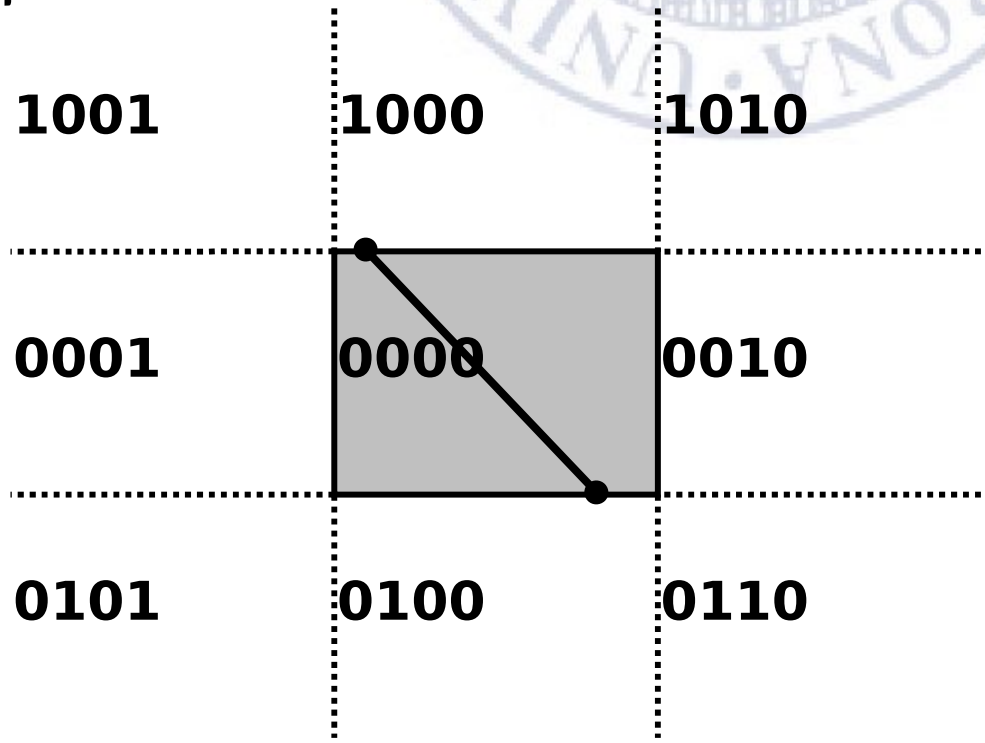
Clipping di un segmento (Cohen-Sutherland)

- Se il risultato dell'AND è nullo:
  - Si individua l'intersezione tra il segmento ed il lato relativo al primo bit discordante tra i codici (bit 1, $y=y_{max}$ in fig.);
  - L'estremo con bit a 1 (in prima posizione nell'esempio) viene sostituito dal nuovo vertice;
  - Si itera il procedimento (in fig., bit 2 discordante, intersezione del segmento con $y=y_{min}$);



Clipping di un segmento (Cohen-Sutherland)

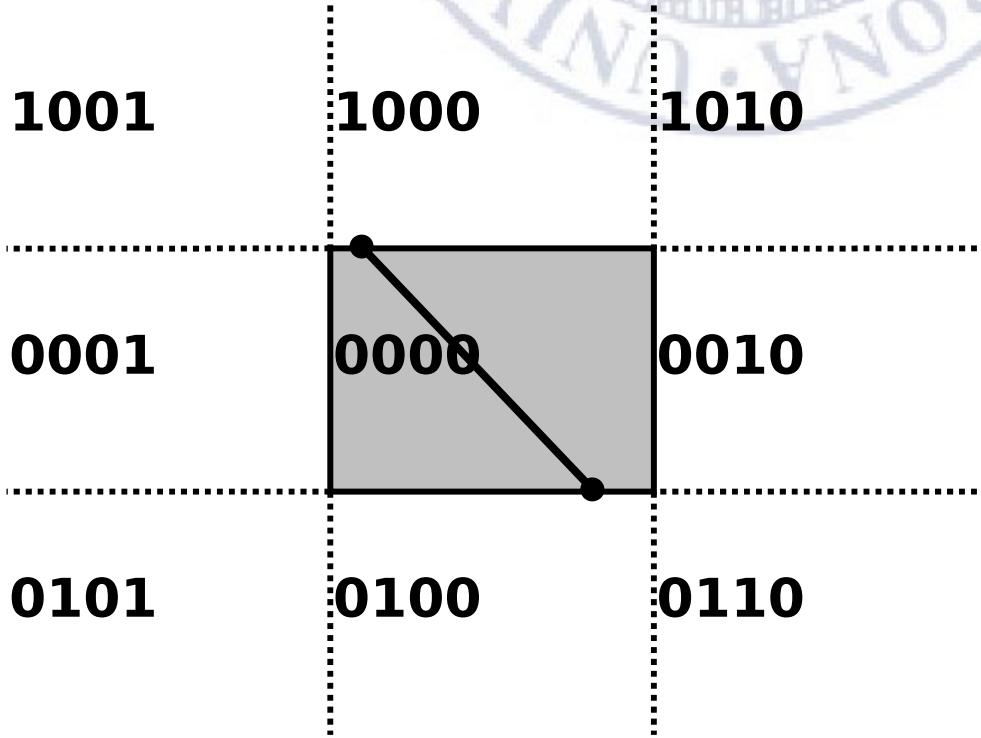
- Se il risultato dell'AND è nullo:
  - Si individua l'intersezione tra il segmento ed il lato relativo al primo bit discordante tra i codici (bit 1, $y=y_{max}$ in fig.);
  - L'estremo con bit a 1 (in prima posizione nell'esempio) viene sostituito dal nuovo vertice;
  - Si itera il procedimento (in fig., bit 2 discordante, intersezione del segmento con $y=y_{min}$);
  - L'estremo con bit a 1 (il bit 2 in fig.) viene sostituito dal nuovo estremo.



Clipping di un segmento (Cohen-Sutherland)

- Ad ogni iterazione si controlla l'eventuale terminazione del processo (OR logico nullo);

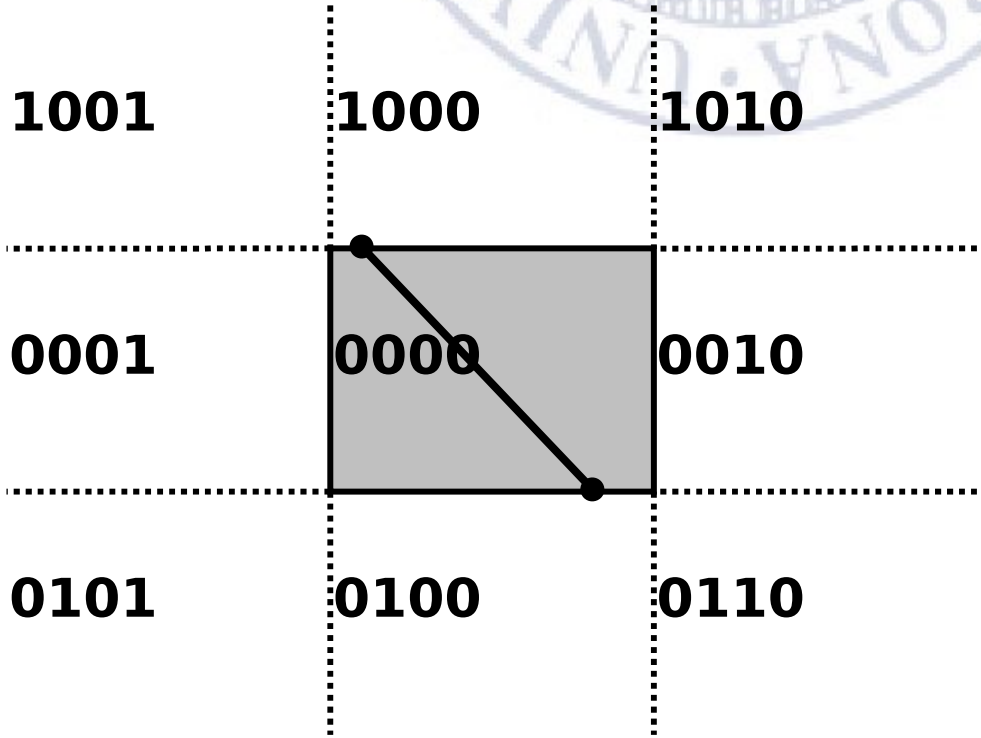
- L'algoritmo rimuove progressivamente le parti esterne; risulta efficiente quando molti dei segmenti da clippare sono completamente esterni al rettangolo di clipping.



Clipping di un segmento (Cohen-Sutherland)

- Ad ogni iterazione si controlla l'eventuale terminazione del processo (OR logico nullo);

- L'algoritmo rimuove progressivamente le parti esterne; risulta efficiente quando molti dei segmenti da clippare sono completamente esterni al rettangolo di clipping.



Clipping di un segmento (Liang-Barsky)

- Il generico segmento AB di estremi (x_a,y_a) e (x_b,y_b) può essere espresso in forma parametrica come:

$$x = x_a + (x_b - x_a)t = x_a + \Delta x t,$$
$$y = y_a + (y_b - y_a)t = y_a + \Delta y t$$

- Per ogni punto (x,y) del segmento interno al rettangolo di clipping valgono le relazioni:

$$x_{min} \leq x_a + \Delta x t \leq x_{max},$$
$$y_{min} \leq y_a + \Delta y t \leq y_{max}.$$

- Che possono essere riscritte nelle 4 disuguaglianze:

$$p_k t \leq q_k \quad k = 1, 2, 3, 4$$

Clipping di un segmento (Liang-Barsky)

- Che possono essere riscritte nelle 4 disuguaglianze:

$$p_k t \leq q_k \quad k = 1, 2, 3, 4$$

- Dove p e q valgono le quantità:

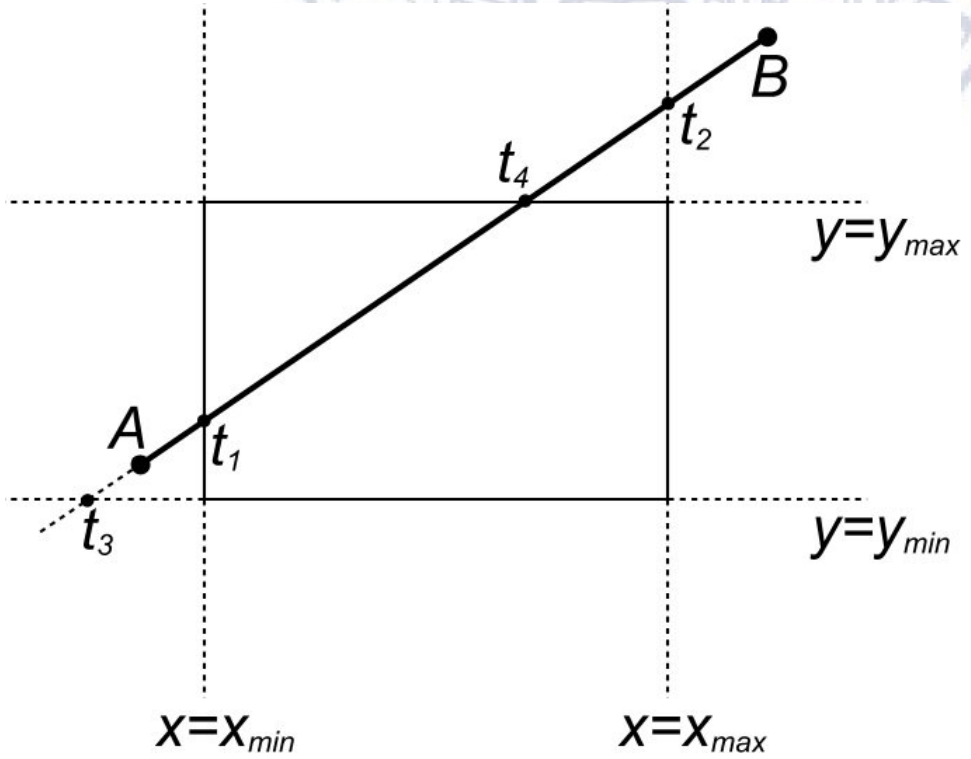
$$\begin{aligned} p_1 &= -\Delta x, \quad & q_1 &= x_a - x_{min}, \\ p_2 &= \Delta x, \quad & q_2 &= x_{max} - x_a, \\ p_3 &= -\Delta y, \quad & q_3 &= y_a - y_{min}, \\ p_4 &= \Delta y, \quad & q_4 &= y_{max} - y_a, \end{aligned}$$

- Relative alle relazioni per il vincolo $x=x_{min}$ (1), $x=x_{max}$ (2), $y=y_{min}$ (3) e $y=y_{max}$ (4).

Clipping di un segmento (Liang-Barsky)

- A(14,19), B(33,32)
- $x_{min}=16, x_{max}=30, y_{min}=18, y_{max}=27$

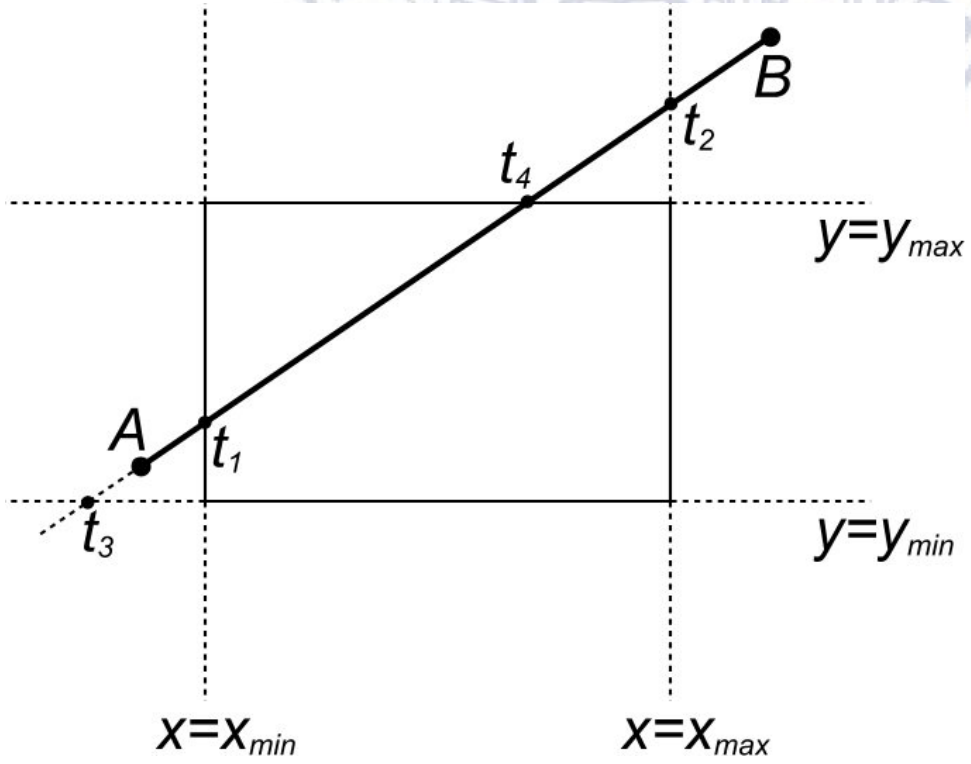
$$
\begin{aligned}
p_1 &= -19, \quad q_1 = -2, \\
p_2 &= 19, \quad q_2 = 16, \\
p_3 &= -13, \quad q_3 = 1, \\
p_4 &= 13, \quad q_4 = 8.
\end{aligned}
$$



- Se $p_k<0$ allora nel muoversi nel verso da A a B si passa da fuori a dentro rispetto al vincolo k;
- Se $p_k>0$ allora nel muoversi nel verso da A a B si passa da dentro a fuori rispetto al vincolo k;
- $t_1=0.105$, $t_2=0.842$, $t_3=-0.077$, $t_4=0.615$

Clipping di un segmento (Liang-Barsky)

- t_1=0.105, t_2=0.842, t_3=-0.077, t_4=0.615
- La parte di AB interna al rettangolo di clipping è individuata da t_e (entrata) e t_u (uscita) dove:
- t_e è il massimo tra 0 (valore minimo di t) ed i valori t_k per cui si entra nella regione di clipping (p_k<0)
- t_e = max(0,t_1,t_3) = 0.105
- t_u è il minimo tra 1 (valore massimo di t) ed i valori t_k per cui si esce dalla regione di clipping (p_k>0)
- t_u = min(1,t_2,t_4) = 0.615

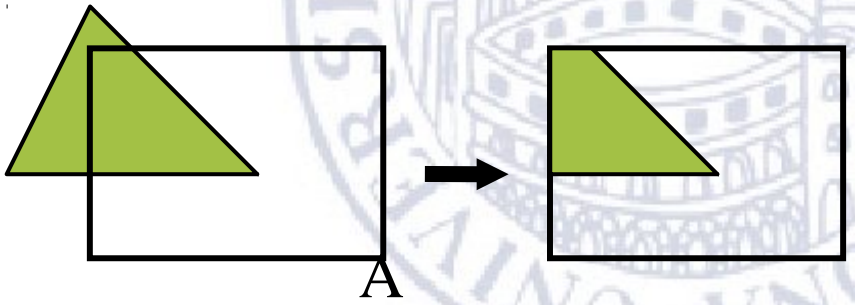


Clipping di un poligono

- Il clipping di un poligono è un'operazione più complessa rispetto al clipping di un segmento per diversi aspetti:

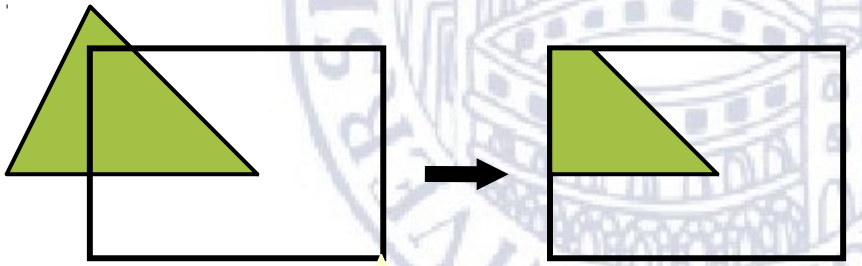
Clipping di un poligono

- Il clipping di un poligono è un'operazione più complessa rispetto al clipping di un segmento per diversi aspetti:
- Dal semplice poligono convesso (A);

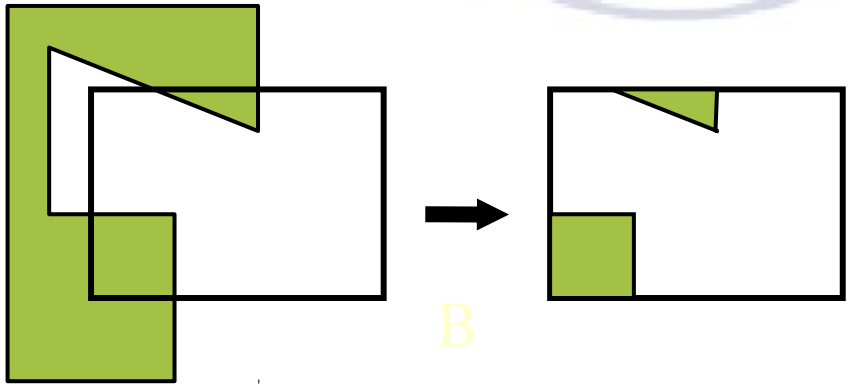


Clipping di un poligono

- Il clipping di un poligono è un'operazione più complessa rispetto al clipping di un segmento per diversi aspetti:
- Dal semplice poligono convesso (A);
- Al poligono concavo che origina più componenti connesse (B);



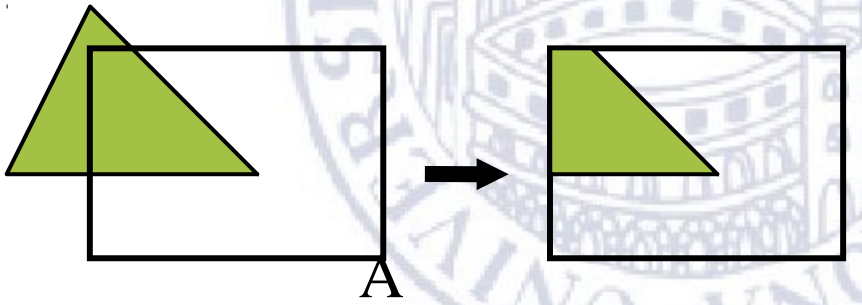
A

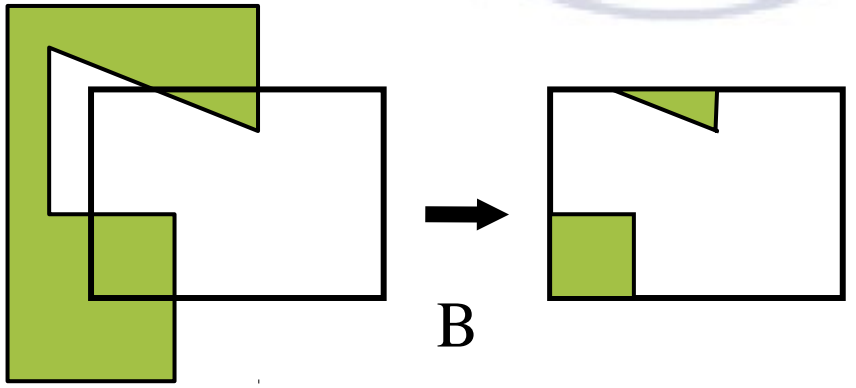


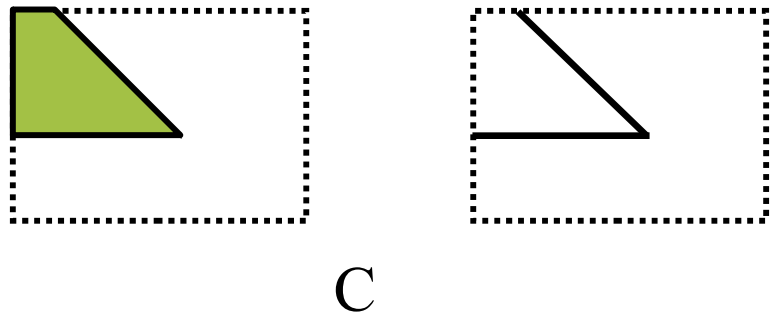
B

Clipping di un poligono

- Il clipping di un poligono è un'operazione più complessa rispetto al clipping di un segmento per diversi aspetti:
- Dal semplice poligono convesso (A);
- Al poligono concavo che origina più componenti connesse (B);
- In ogni caso il risultato consta di uno o più poligoni e non solo segmenti sconnessi (C).

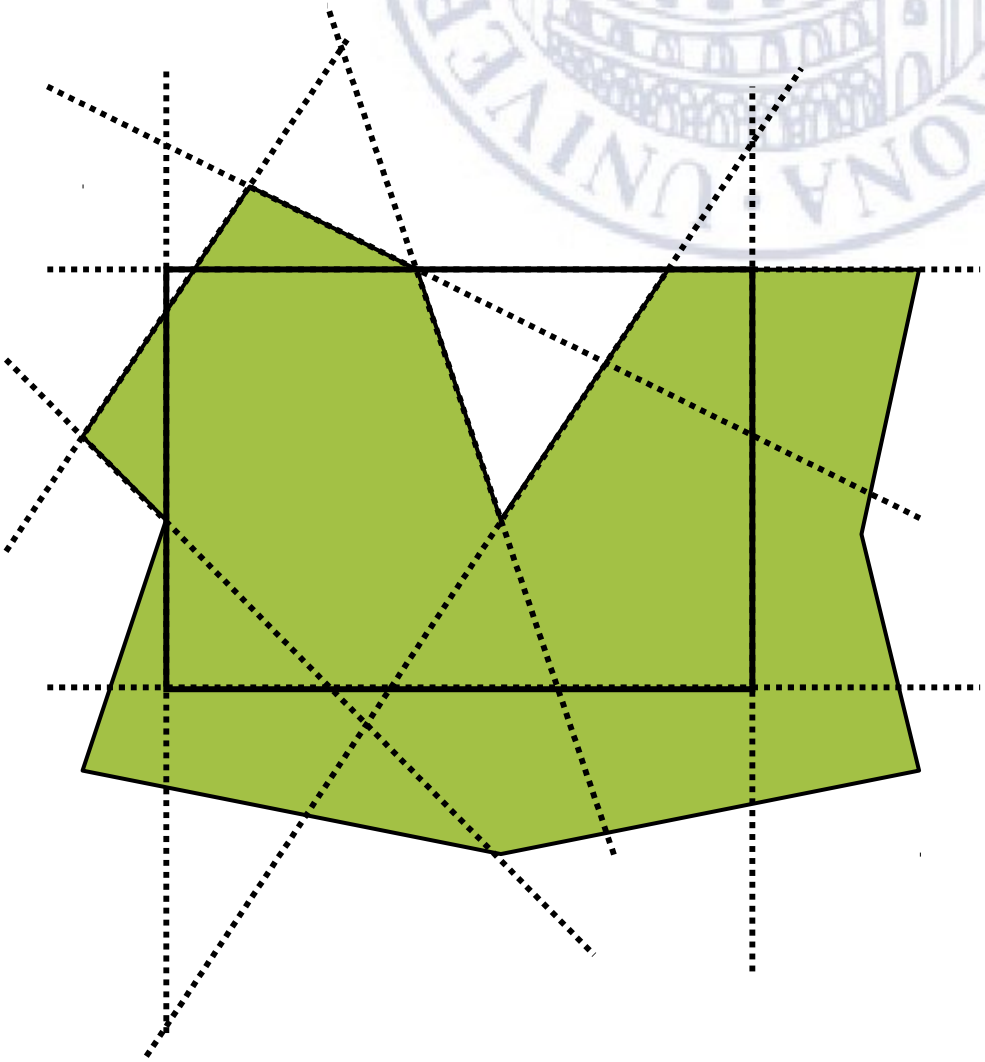






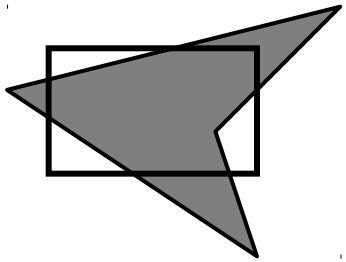
Clipping di un poligono

- L'approccio diretto consiste nel confrontare ogni lato del poligono con le 4 rette che delimitano il rettangolo di clipping;
- Questo approccio implica l'esecuzione di operazioni costose (la determinazione di intersezioni) e spesso inutili.

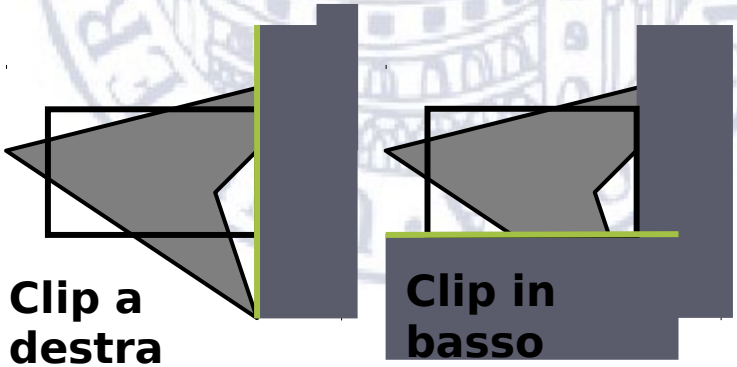


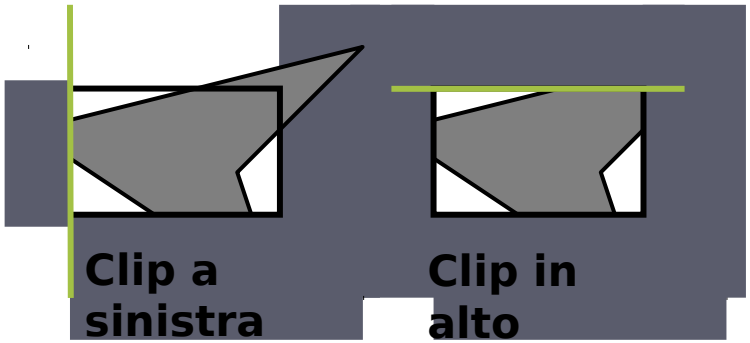
Clipping di un poligono (Sutherland-Hodgman)

- Approccio divide et impera;
- Il problema è ricondotto al clipping di un poligono generico rispetto ad una retta;
- La procedura è applicata sequenzialmente alle 4 rette che definiscono il rettangolo di clipping.



Poligono originale



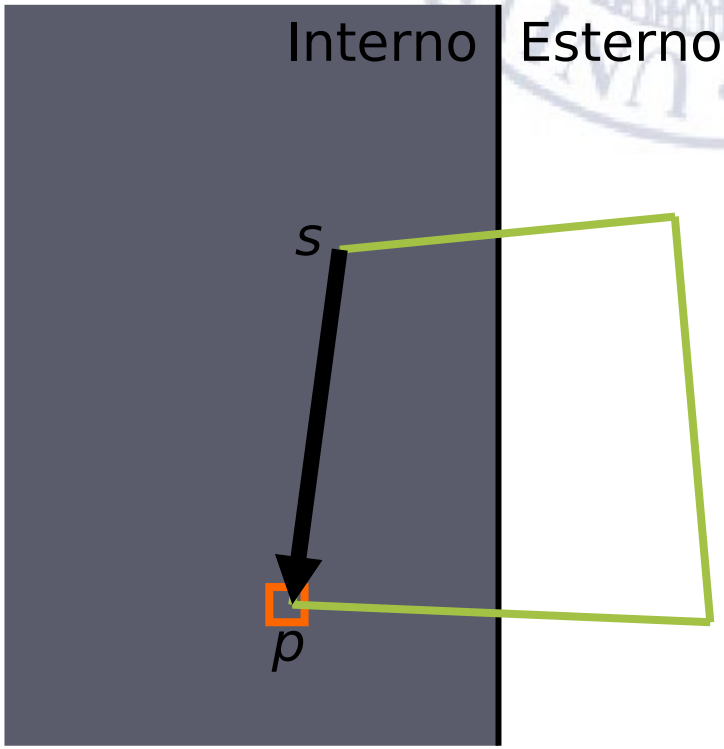


Clipping di un poligono (Sutherland-Hodgman)

- In ingresso la sequenza dei vertici $v_1, v_2, \ldots, v_n$ che definiscono gli n spigoli ($n-1$ da v_i a v_{i+1}, $(i=1, 2, \ldots, n)$ e lo spigolo da v_n a v_1);
- Ad ogni iterazione il generico vincolo (ad esempio $x=x_{min}$) individua un semipiano interno (corrispondente alla finestra di clipping) ed uno esterno;
- Il confronto si effettua visitando il poligono dal vertice v_1 al vertice v_n e quindi di nuovo a v_1;
- Ad ogni passo dell'algoritmo si analizza la relazione esistente tra due vertici successivi sul poligono e la retta di clipping (4 casi possibili);
- Ad ogni iterazione, i vertici inseriti nella lista di output rappresentano i dati in ingresso per l'iterazione successiva.

Clipping di un poligono (Sutherland-Hodgman)

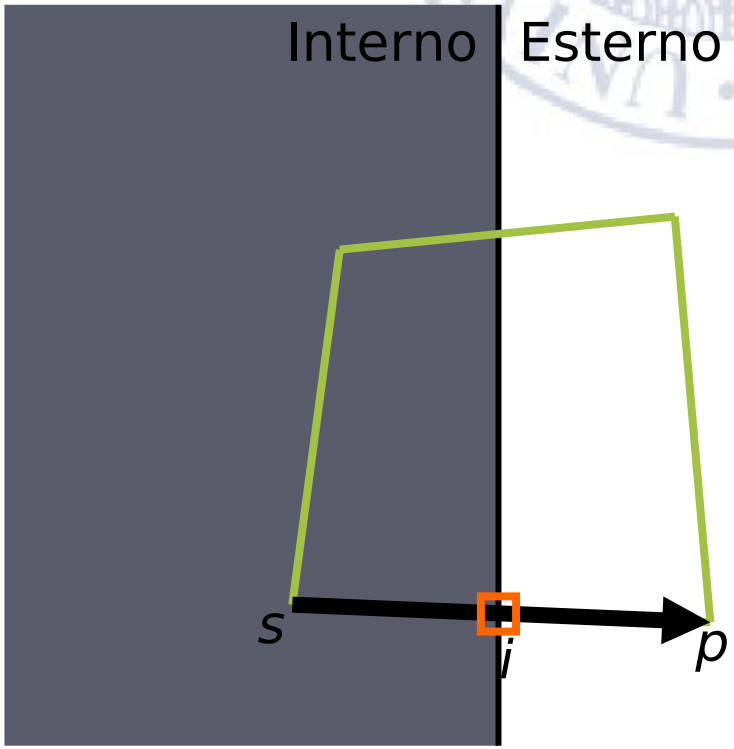
- Caso 1: i vertici *s* e *p* giacciono sul semipiano interno; *p* è aggiunto alla lista di vertici di output.



Caso 1

Clipping di un poligono (Sutherland-Hodgman)

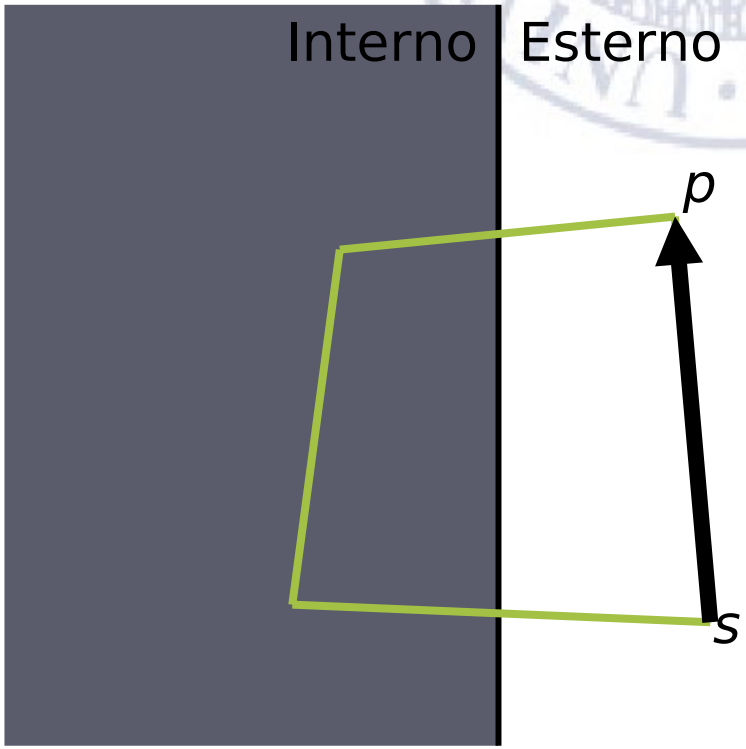
- Caso 1: i vertici *s* e *p* giacciono sul semipiano interno; *p* è aggiunto alla lista di vertici di output;
- Caso 2: *s* è interno, *p* è esterno; l'intersezione *i* dello spigolo con il vincolo è aggiunta alla lista di output.



Caso 2

Clipping di un poligono (Sutherland-Hodgman)

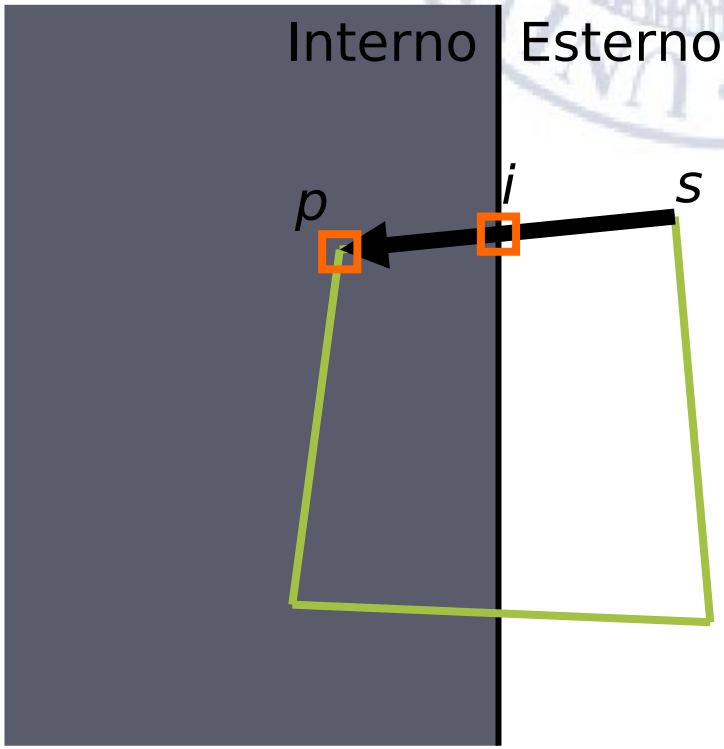
- Caso 1: i vertici *s* e *p* giacciono sul semipiano interno; *p* è aggiunto alla lista di vertici di output;
- Caso 2: *s* è interno, *p* è esterno; l'intersezione *i* dello spigolo con il vincolo è aggiunta alla lista di output;
- Caso 3: *s* e *p* giacciono sul semipiano esterno; nessuna azione.



Caso 3

Clipping di un poligono (Sutherland-Hodgman)

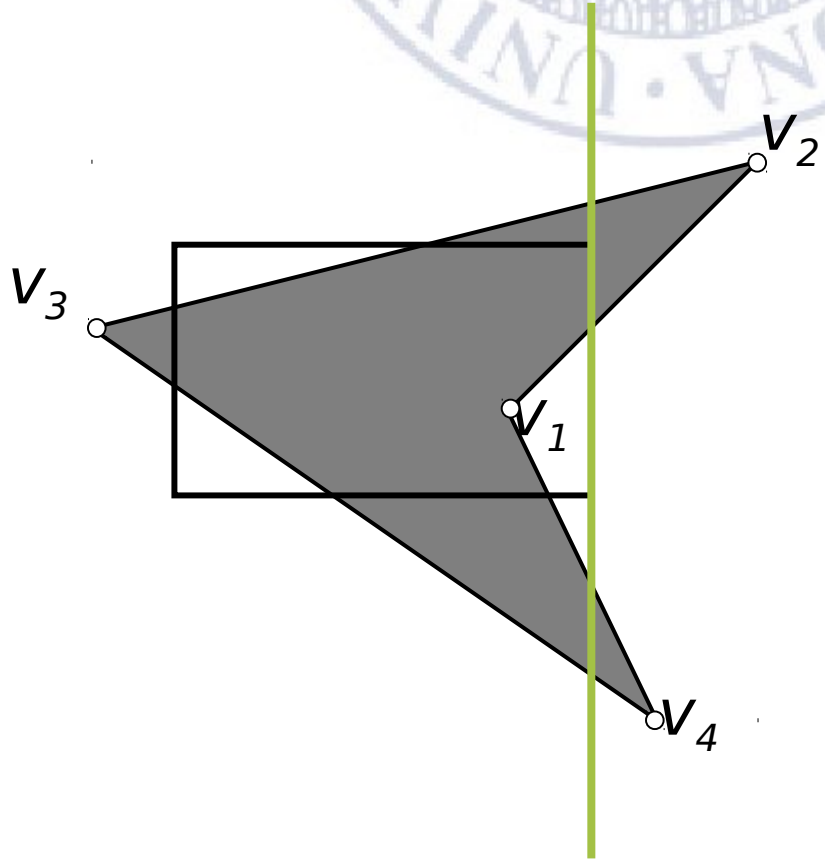
- Caso 1: i vertici *s* e *p* giacciono sul semipiano interno; *p* è aggiunto alla lista di vertici di output;
- Caso 2: *s* è interno, *p* è esterno; l'intersezione *i* dello spigolo con il vincolo è aggiunta alla lista di output;
- Caso 3: *s* e *p* giacciono sul semipiano esterno; nessuna azione;
- Caso 4: *s* è esterno, *p* è interno; L'intersezione *i* ed il vertice *p* in output.



Caso 4

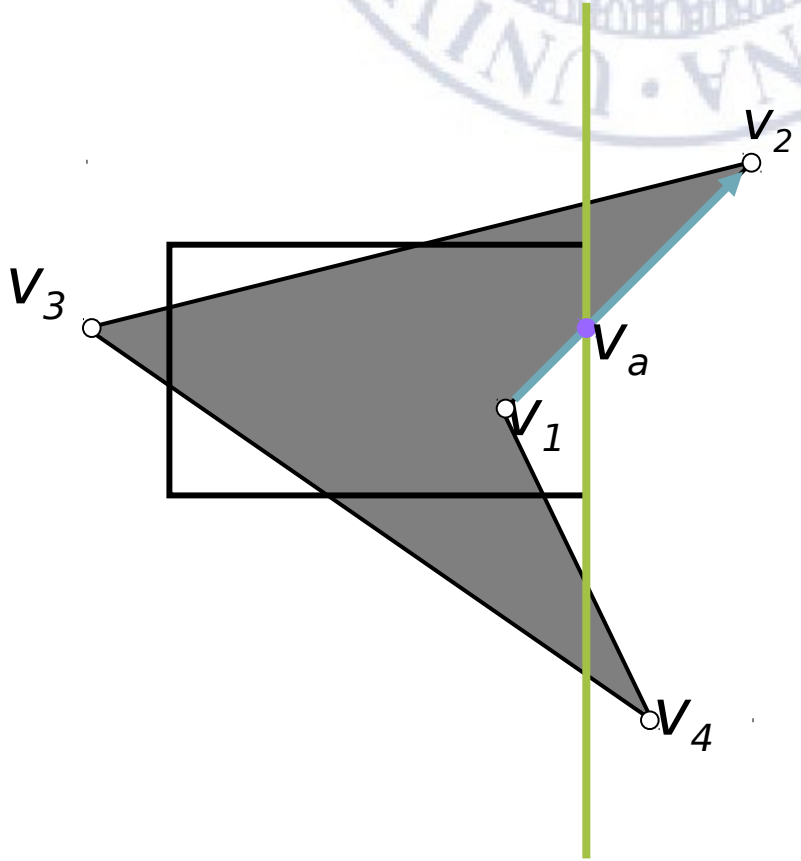
Clipping di un poligono (Sutherland-Hodgman)

- Esempio:
- In: v_1, v_2, v_3, v_4
- Out: /



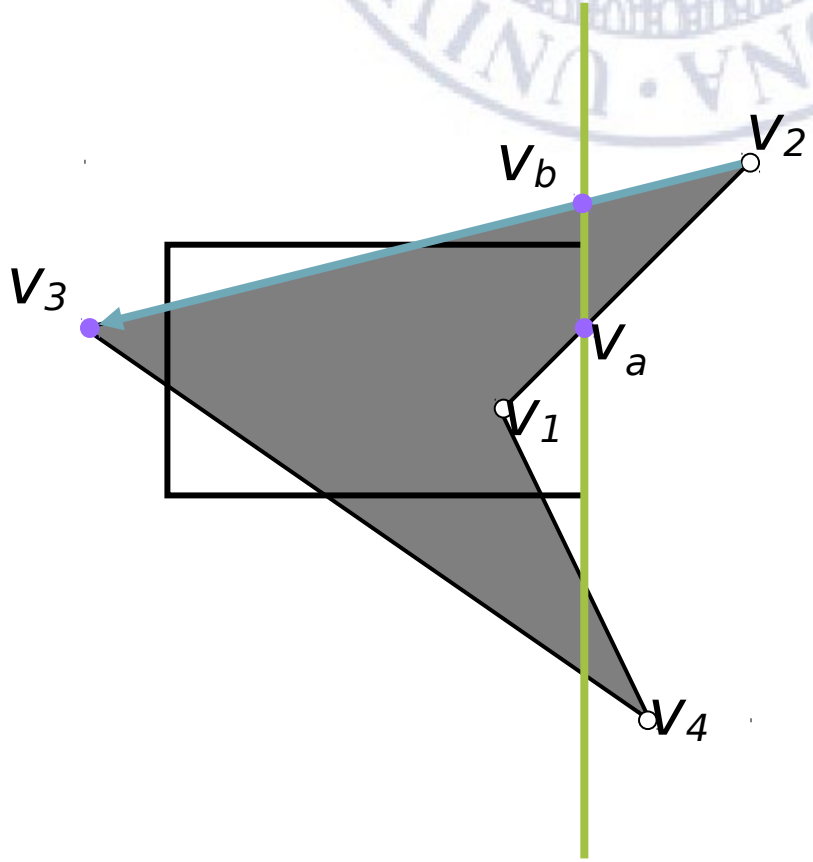
Clipping di un poligono (Sutherland-Hodgman)

- Esempio:
- In: v_1, v_2, v_3, v_4
- Caso 2, spigolo uscente
- Out: v_a



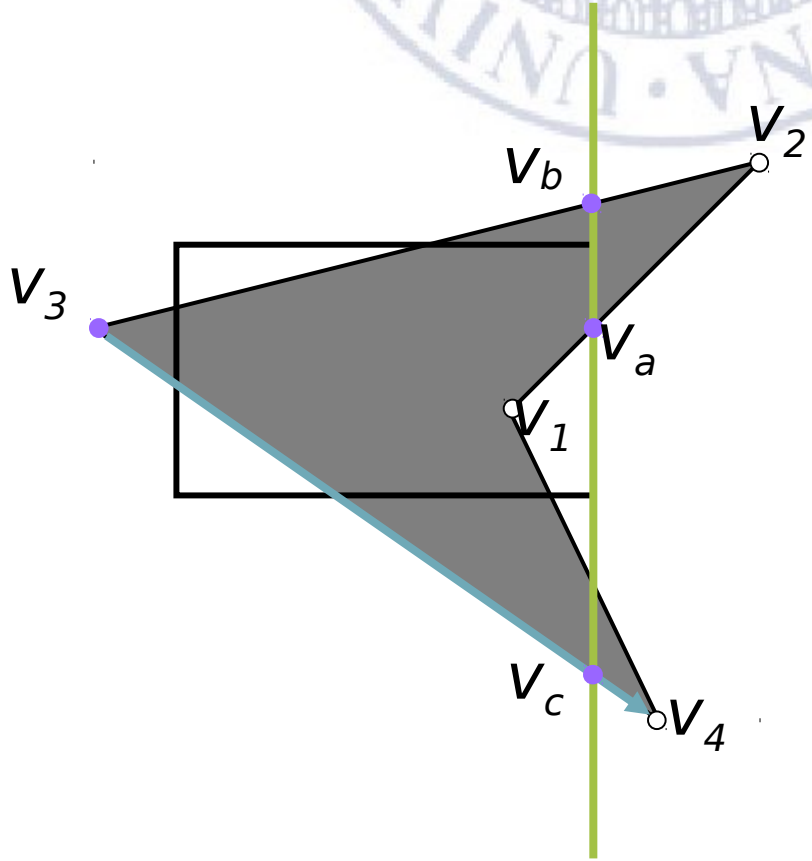
Clipping di un poligono (Sutherland-Hodgman)

- Esempio:
- In: v_1, v_2, v_3, v_4
- Caso 4, spigolo entrante
- Out: v_a, v_b, v_3



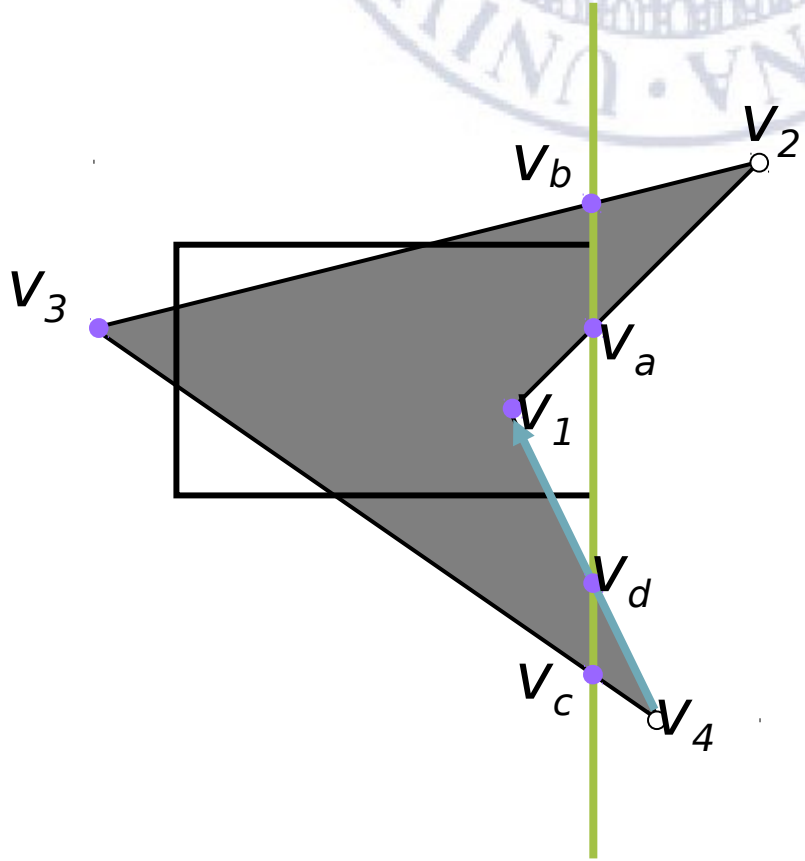
Clipping di un poligono (Sutherland-Hodgman)

- Esempio:
- In: v_1, v_2, v_3, v_4
- Caso 2, spigolo uscente
- Out: v_a, v_b, v_3, v_c



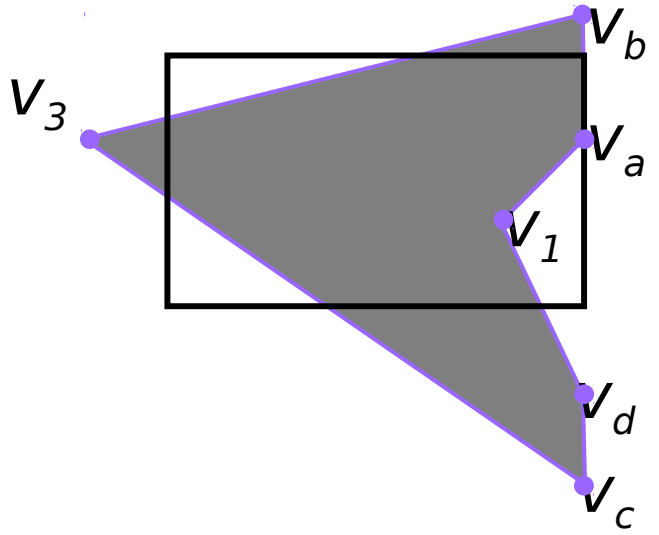
Clipping di un poligono (Sutherland-Hodgman)

- Esempio:
- In: v_1, v_2, v_3, v_4
- Caso 4, spigolo entrante
- Out: $v_a, v_b, v_3, v_c, v_d, v_1$



Clipping di un poligono (Sutherland-Hodgman)

- Nei casi in cui il clipping dia luogo a più componenti è possibile che si generino falsi spigoli sovrapposti ai bordi del rettangolo di clipping;
- L'algoritmo deve prevedere una fase di post-elaborazione per la rimozione di tali spigoli.



Rimozione superfici nascoste (HSR)

- Gli oggetti della scena sono generalmente opachi;
- Gli oggetti più vicini all'osservatore possono nascondere (occludere) la vista (totale o parziale) di oggetti più lontani;

Rimozione superfici nascoste (HSR)

- Il problema della rimozione delle superfici nascoste (HSR, Hidden surface removal) consiste nel determinare le parti della scena non visibili dall'osservatore;
- La rimozione delle superfici nascoste non è solo dipendente dalla disposizione degli oggetti nella scena ma anche dalla relazione esistente tra oggetti e posizione dell'osservatore.

HSR

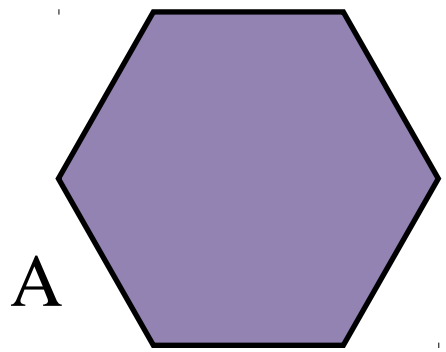
- Gli algoritmi per la rimozione delle superfici nascoste si possono classificare in:
  - gli algoritmi che operano in object-space determinano, per ogni primitiva geometrica della scena, le parti della primitiva che non risultano oscurate da altre primitive nella scena. Gli algoritmi operano nello spazio di definizione delle primitive
  - gli algoritmi che operano in image-space determinano, per ogni punto "significativo" del piano di proiezione (ogni pixel del piano del piano immagine), la primitiva geometrica visibile "attraverso" quel punto. Gli algoritmi operano nello spazio immagine della scena proiettata.

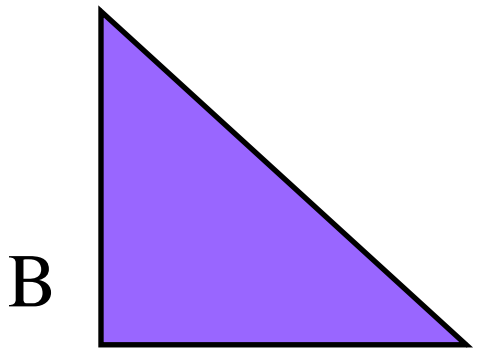
Applicazione o Geometry subsystem

Raster subsystem

HSR: Object-space

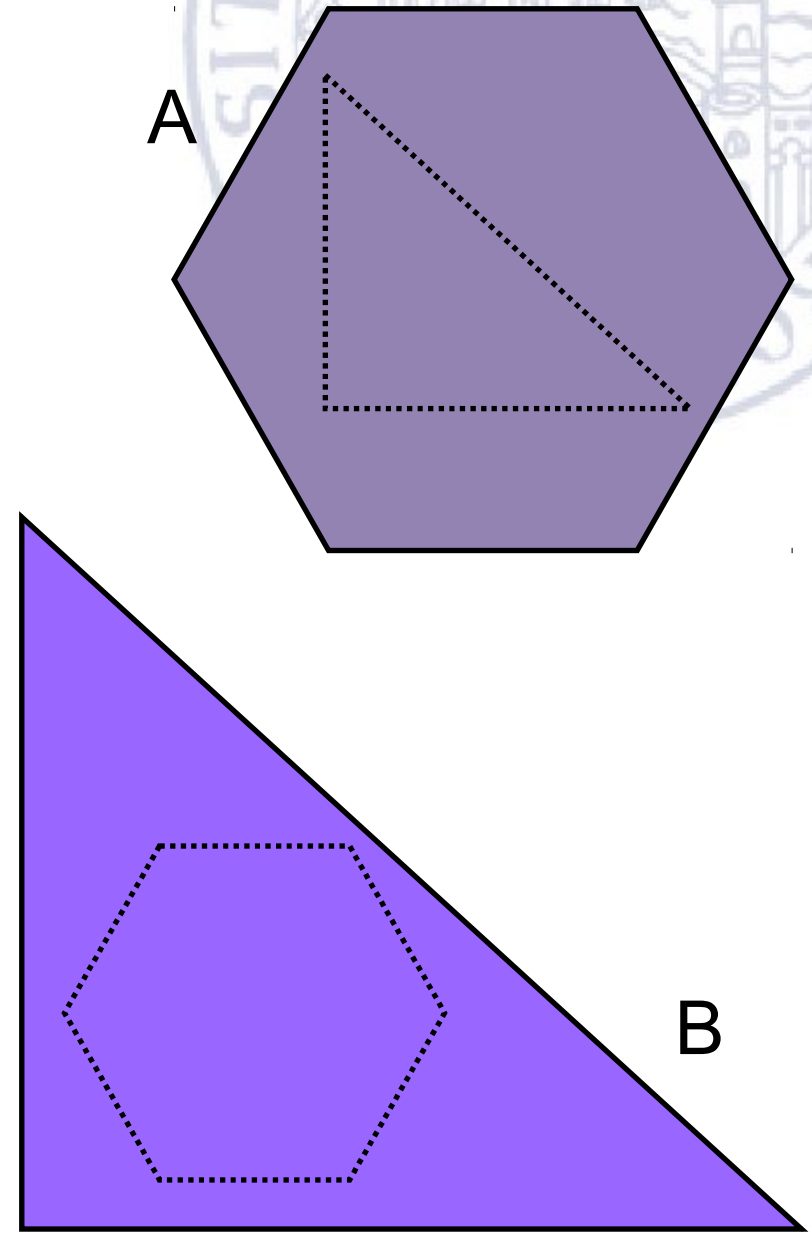
- Nell'ipotesi di una scena 3D composta da *k* primitive geometriche planari ed opache, si può derivare un generico algoritmo di tipo object-space analizzando gli oggetti a coppie;
- Fissato un punto di vista, le relazioni spaziali di due primitive geometriche A e B possono essere:
  - A [B] oscura B [A]; solo A [B] è visualizzata;
  - A e B sono completamente visibili; entrambe le primitive sono visualizzate;
  - A [B] occlude parzialmente B [A]: è necessario individuare le parti visibili di B [A].





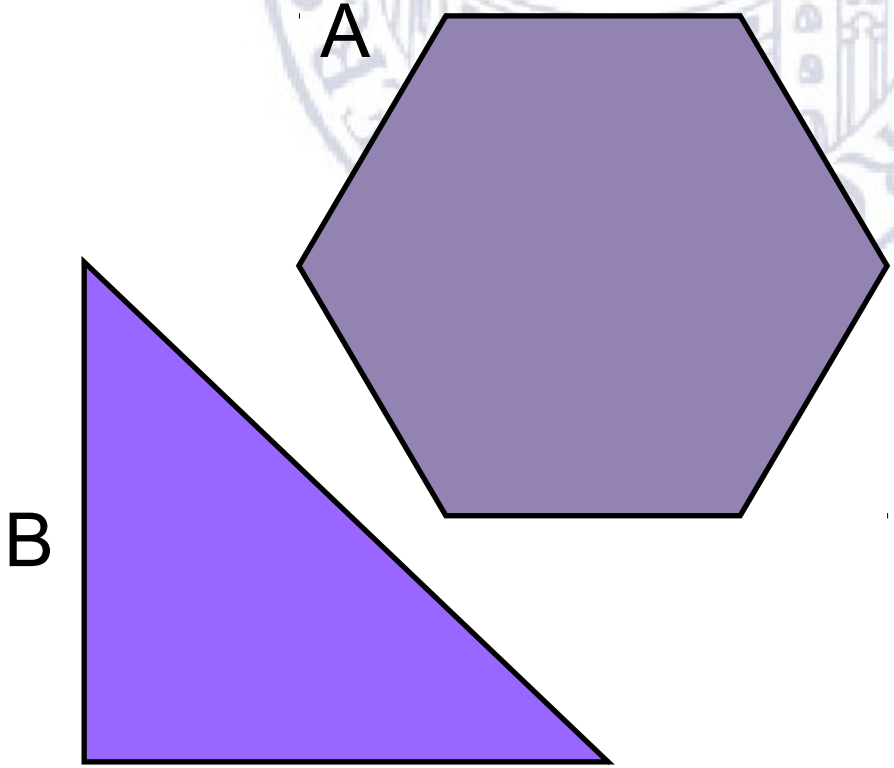
HSR: Object-space

- A oscura B: tutti i punti di A sono più vicini all'osservatore di tutti i punti di B e la proiezione di B sul piano di vista ricade all'interno della proiezione di A. Visualizza A;
- B oscura A: tutti i punti di B sono più vicini all'osservatore di tutti i punti di A e la proiezione di A sul piano di vista ricade all'interno della proiezione di B. Visualizza B



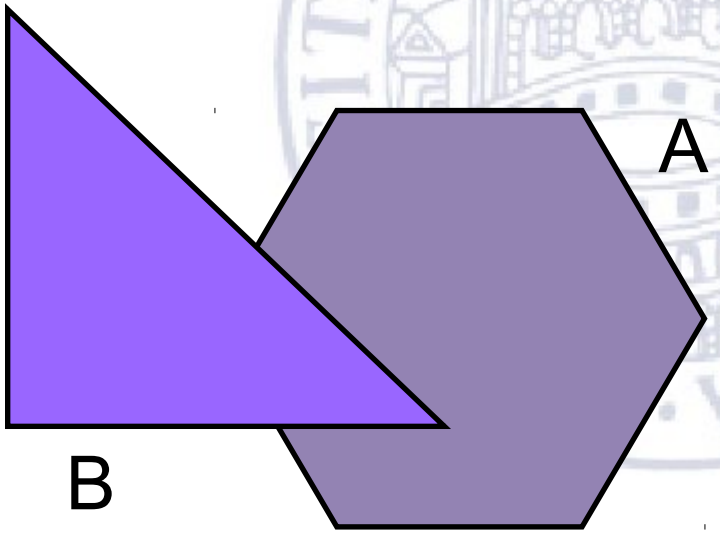
HSR: Object-space

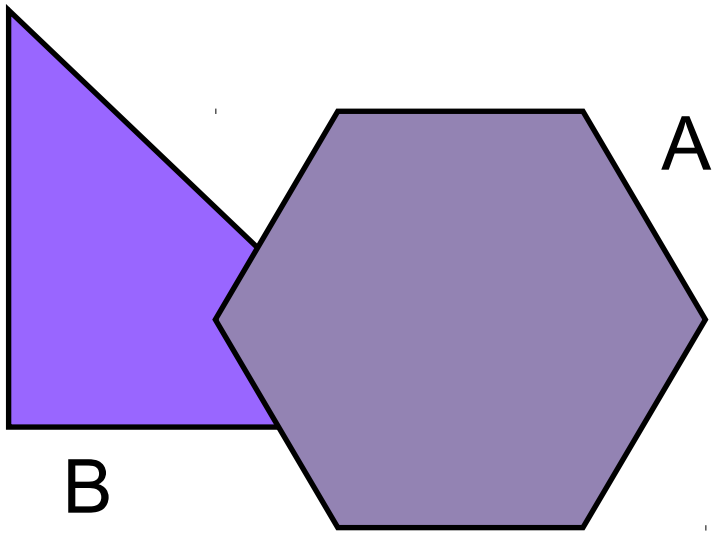
- A e B non si occludono: le due proiezioni di A e B sul piano di vista sono disgiunte;
- Visualizza A e B.



HSR: Object-space

- A e B si oscurano parzialmente: l'intersezione tra le due proiezioni di A e B sul piano di vista è non nulla e diversa sia dalla proiezione di A che da quella di B.
- Individua le parti visibili di ciascuna primitiva.



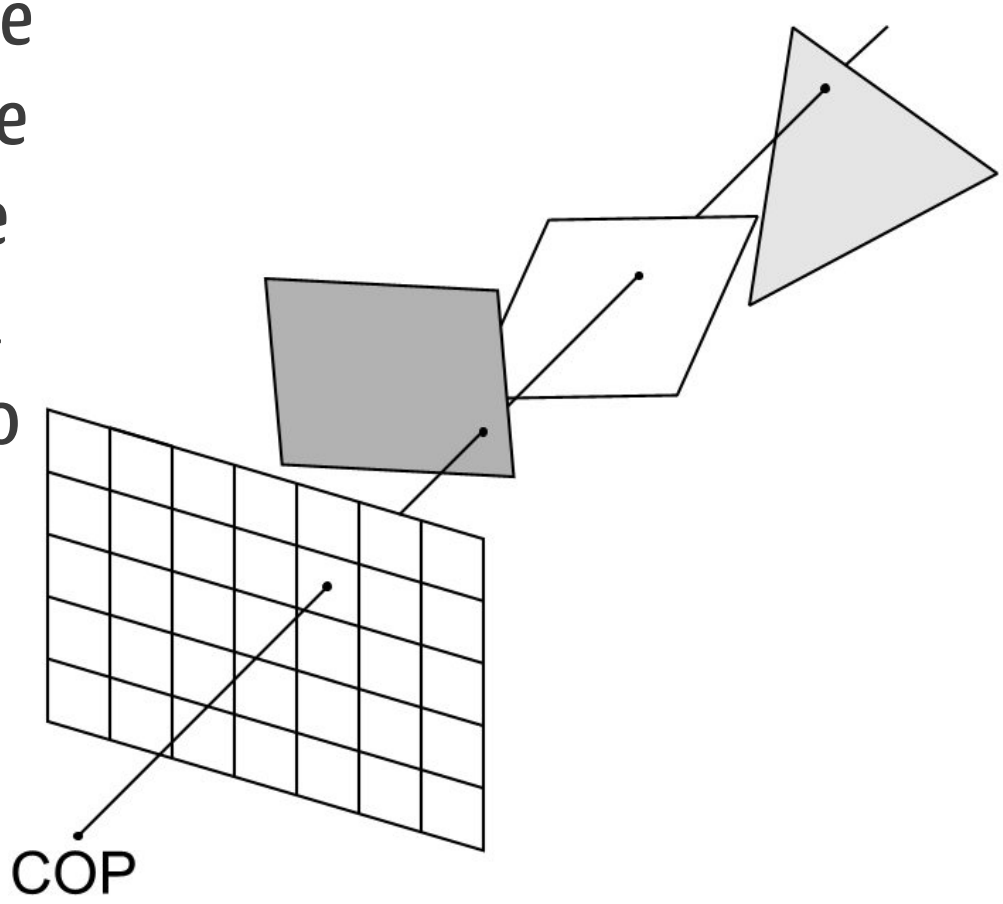


HSR: Object-space

- Un possibile algoritmo risolutivo:
  - Proiettare le *k* primitive geometriche;
  - Al generico passo analizzare la *i-esima* primitiva (i=1, …, k-1) con le rimanenti $k - i$ in modo da individuare le parti visibili.
- La complessità dell'approccio object-space risulta di ordine $O(k^2)$
- L'approccio object-space è consigliabile solo quando le primitive nella scena sono relativamente poche.

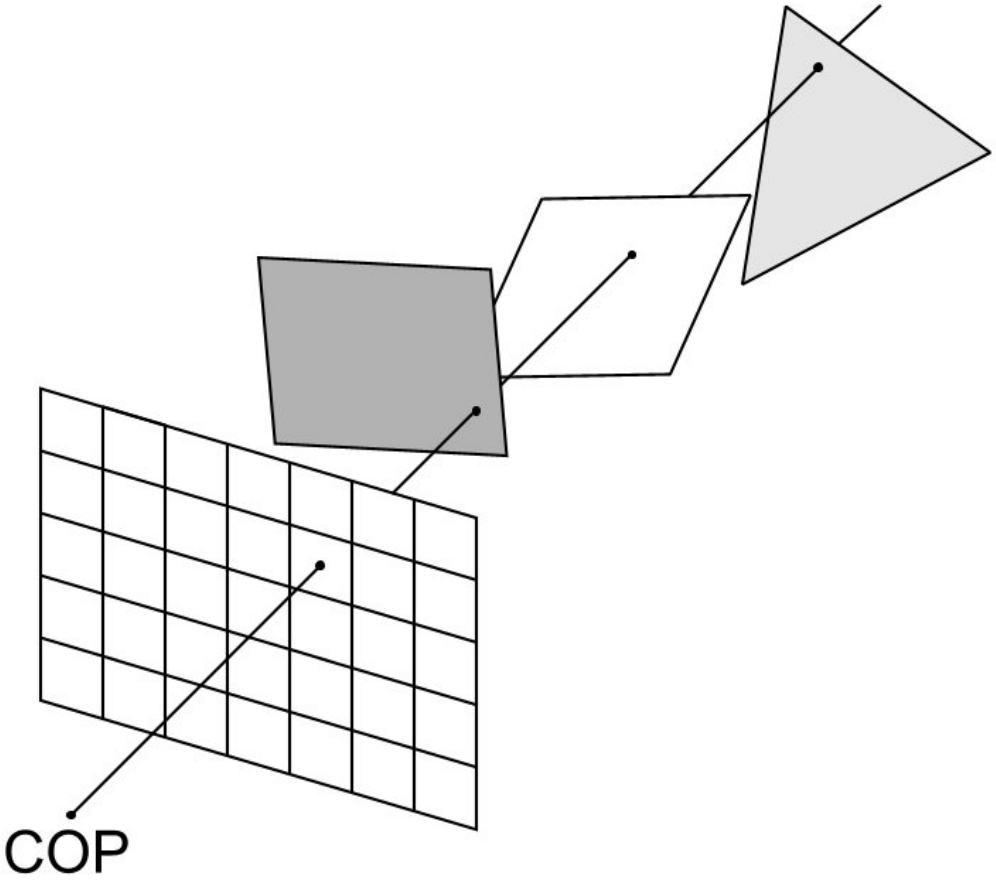
HSR: Image-space

- Per ogni pixel del piano immagine si considera una semiretta avente origine nel centro di proiezione e passante per il pixel in esame. La semiretta attraversa la scena fino a colpire una primitiva o a perdersi sul fondo.



HSR: Image-space

- Per ogni primitiva si calcola l'intersezione della semiretta con il piano di appartenenza:
  - Se l'intersezione è interna alla primitiva si memorizza la distanza del punto di intersezione dal piano di vista;
  - Se l'intersezione è esterna viene immediatamente scartata.
- Tra le distanze accumulate si sceglie la minore (l'intersezione più vicina al centro di proiezione) e si attribuisce



HSR: Image-space

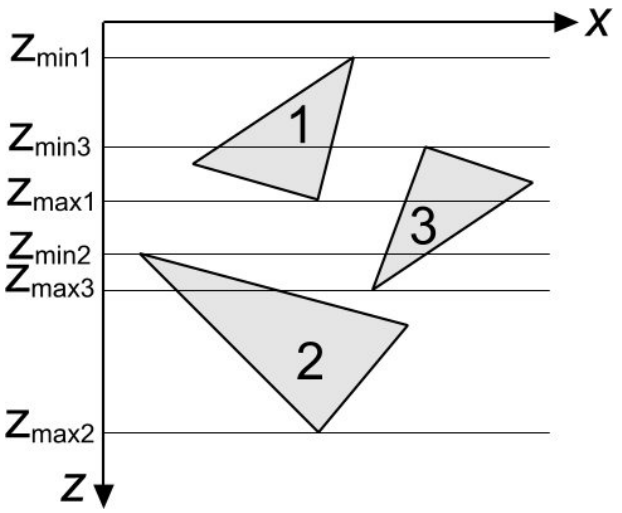
- L'operazione fondamentale dell'approccio image-space è il calcolo delle intersezioni tra semirette e piani di appartenenza delle primitive (per ogni semiretta al più k intersezioni);
- Anche se per un display n x m, questa operazione deve essere eseguita n x m x k volte, la complessità risulta comunque di ordine O(k)
- Sia nell'approccio object-space che in quello image-space la rimozione delle superfici nascoste è riconducibile ad un problema di ordinamento (in profondità).

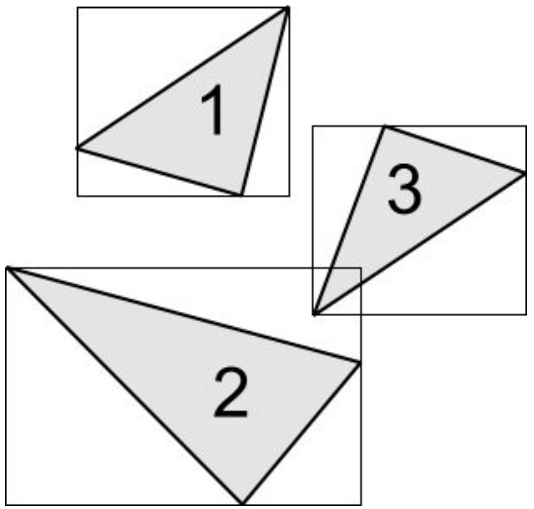
HSR: Aumentare l'efficienza

- coerenza
  - Coerenza a livello di oggetto;
  - Coerenza in profondità (parti adiacenti di una stessa primitiva sono "vicine" in profondità).
- Trasformazioni prospettiche e parallele
  - Il punto $P_1(x_1,y_1,z_1)$ oscura $P_2=(x_2,y_2,z_2)$ se P_1 e P_2 sono sullo stesso proiettore;
  - Nella proiezione parallela la verifica è $x_1=x_2$ e $y_1=y_2$;
  - Per una proiezione prospettica occorre verificare che $x_1/z_1=x_2/z_2$ e $y_1/z_1=y_2/z_2$;
  - Può convenire applicare una trasformazione geometrica globale in modo tale che la proiezione parallela della scena trasformata sia uguale alla proiezione prospettica non trasformata.

HSR: Aumentare l'efficienza

- Extent, bounding box e bounding volume
  - Extent 1D, bounding box 2D e bounding volume 3D si rivelano fondamentali per l'ordinamento. Se non c'è sovrapposizione tra i bounding box allora non occorre confrontare le relative primitive (condizione necessaria ma non sufficiente!).





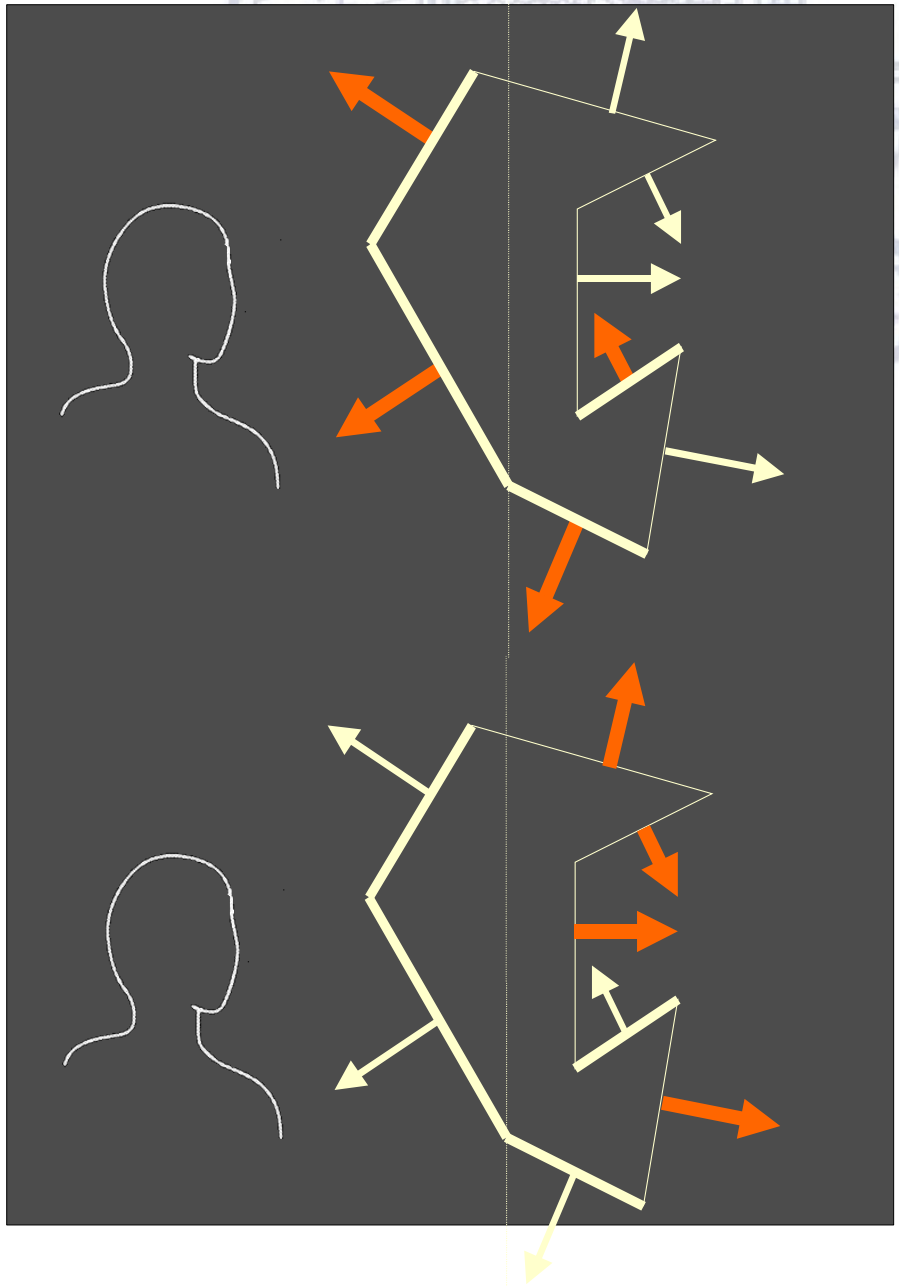
Back-face culling

- Back-face culling (Eliminazione delle facce posteriori)
  - Se gli oggetti della scena sono rappresentati da poliedri solidi chiusi (cioè le facce poligonali dei poliedri delimitano completamente i volumi dei solidi);
  - Se ogni faccia poligonale è stata modellata in modo tale che la normale ad essa sia diretta verso l'esterno del poliedro di appartenenza;
  - Se nessuna parte del poliedro viene tagliata dal front clipping plane
  - ...

Back-face culling

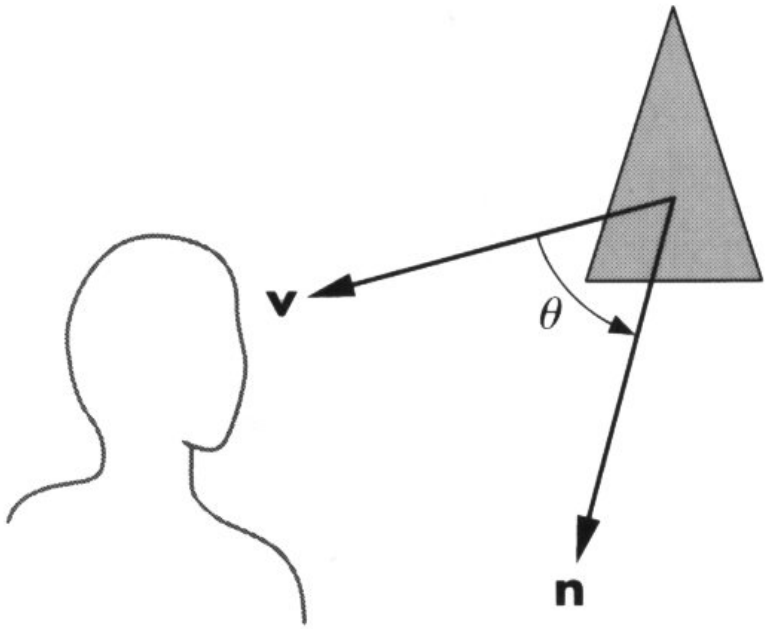
- Nelle ipotesi precedenti...
- Le facce la cui normale forma angoli superiori a ±90° con la direzione di vista possono essere visibili;

- Le facce la cui normale forma angoli inferiori a ±90° con la direzione di vista certamente non sono visibili;



Back-face culling

- Per ridurre il carico di lavoro richiesto per la rimozione delle superfici nascoste può essere quindi opportuno eliminare inizialmente tutti le primitive geometriche la cui normale è orientata verso il semispazio opposto all'osservatore, non visibile all'osservatore;
- Indicato con θ l'angolo tra la normale e l'osservatore, la primitiva in esame deve essere rimossa se $-90^\circ \leq \theta \leq 90^\circ$, cioè se $\cos\theta \geq 0$.
- Invece di calcolare la quantità $\cos\theta$ possiamo valutare il prodotto scalare $n \cdot v \geq 0$



Back-face culling

- Se l'operazione è eseguita in coordinate normalizzate di vista (dopo aver eseguito la proiezione) la determinazione delle facce back-facing si riduce ad un controllo del segno della coordinata z delle normali: ad un segno positivo corrispondono facce front-facing, ad un segno negativo facce back-facing;
- Questo procedimento (detto back-face culling) consente, in media, di dimezzare il tempo necessario per il rendering di oggetti solidi dato che, sempre in media, circa metà delle facce di un poliedro sono back-facing e quindi la loro visualizzazione sarebbe comunque inutile.

HSR: Aumentare l'efficienza

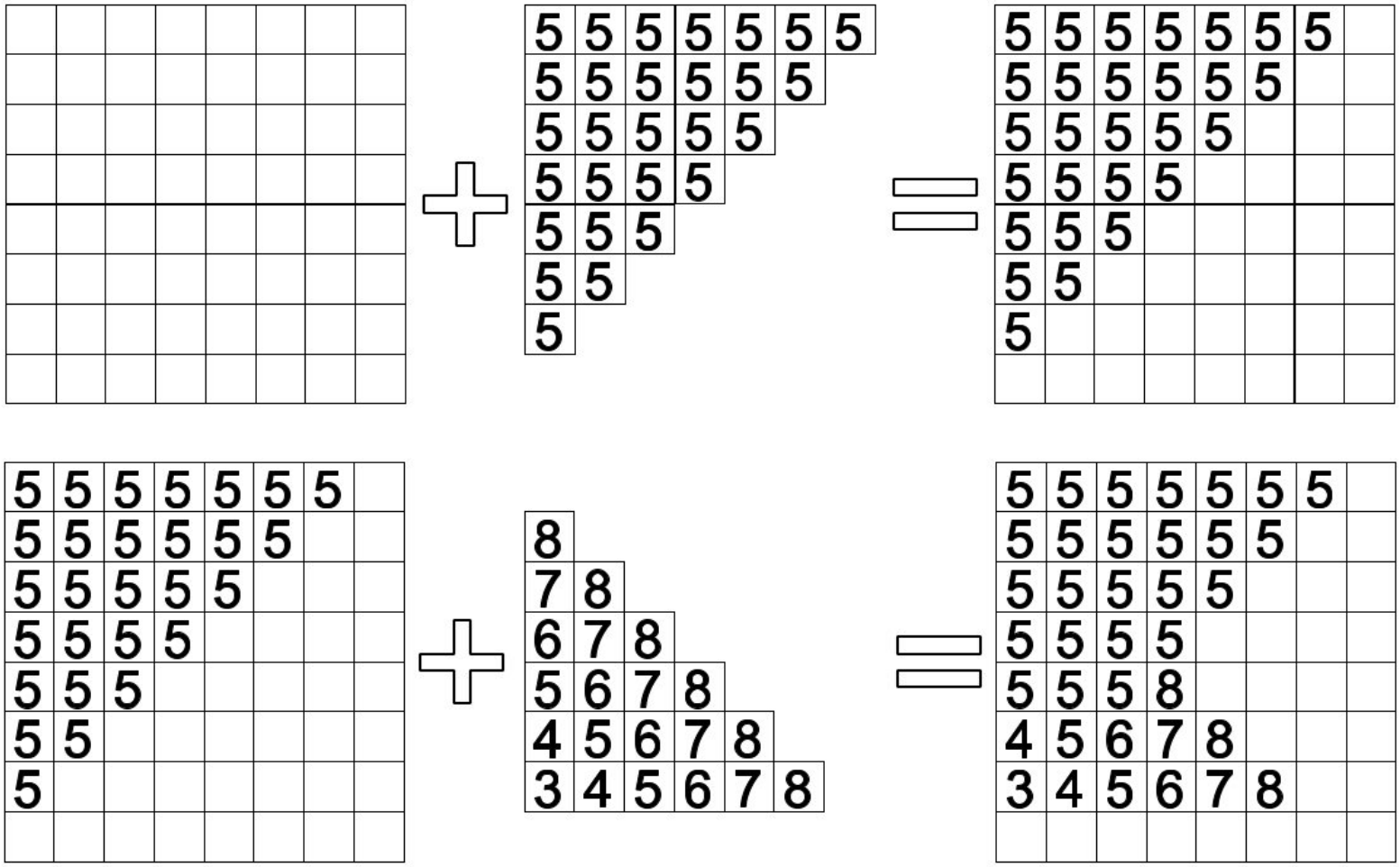
- Suddivisione spaziale
  - Riorganizzare gli oggetti della scena (o le loro proiezioni sul piano immagine) in gruppi coerenti dal punto di vista spaziale;
  - Si effettua di solito mediante griglie 2D o 3D che servono a limitare i confronti in profondità tra le primitive della scena.

HSR: L'algoritmo *z-buffer*

- Durante la fase di rasterizzazione delle primitive si determina, per ogni pixel (x,y) su cui la primitiva viene mappata, la profondità della primitiva in quel punto. La rasterizzazione avviene dopo la proiezione sul piano di vista, nello spazio 3D NDC;
- Se la profondità z in (x,y) è inferiore alla profondità corrente memorizzata nello z-buffer allora (x,y) assume z come profondità corrente ed il pixel (x,y) nel frame buffer assume il valore colore della primitiva in esame.

HSR: L'algoritmo *z-buffer*

- Esempio



HSR: L'algoritmo *z-buffer*

- Lo z-buffer ha la stessa risoluzione del frame buffer ed ogni cella ha dimensione sufficiente per memorizzare la informazioni di profondità alla risoluzione richiesta (di solito 24 bit);
- Ogni elemento dello z-buffer è inizializzato al valore della distanza massima dal centro di proiezione;
- Non è richiesto alcun ordine preventivo tra le primitive geometriche;
- Generalmente implementato firmware;
- Complessità pressoché costante (ad un aumento delle primitive corrisponde in genere una diminuzione della superficie coperta).

HSR: L'algoritmo *z-buffer*

- L'algoritmo z-buffer è un algoritmo di tipo image-space, basato su una logica molto semplice e facile da realizzare;
- Lavora in accoppiamento con l'algoritmo di scan conversion (rasterizzazione, disegno, delle primitive geometriche sulla memoria di quadro) ed ha bisogno, come struttura dati di supporto, di un'area di memoria, il depth buffer, delle stesse dimensioni del frame buffer
- Per ogni posizione (x,y) della vista che si genera, il frame buffer contiene il colore assegnato a quel pixel, il depth buffer la profondità del punto corrispondente sulla primitiva visibile.

Riferimenti

- Scateni cap. 5